Sikkerhedsmodellering af forbrugerelektronik

Security Modelling of Consumer Electronics

Bachelorprojekt i Softwareteknologi – F14

Daniel Tolboe Handler, s113446

Rasmus Lau Petersen, s113449

Vejleder: Christian D. Jensen

Afleveret 18. juli 2014

Bachelorprojekt

Danmarks Tekniske Universitet

Security Modelling of Consumer Electronics

Abstract

Forfatter: Fælles

The development of modern consumer electronics moves towards products, which incorporates features previously contained in PC's and smartphones. Standard products like TV's, fridges and vacuum cleaners are equipped with an internet connection to be *smart* and act together. This development is known as the "Internet of Things" and have a potential to change our everyday lives.
However, this happens at a cost: It increases the vulnerability of ourselves and our products. On regular internet mediated products like laptops and tablets, a certain level of security is implemented and maintained by the manufacturer, but as their functions are propagated to daily-life products, can we assume, that the manufacturer of a smart fridge or a Smart TV has thought adequately about security? Or is it even feasible for them to do so in a highly competitive market with a high replacement rate on the products?

The aim for this project was to make a guide to a manufacturer of internet equipped and *smart* consumer electronics using a self-developed methodology, which can identify possible security problems in these products the best way possible in order to counter the concern given above. The issues were to make this guide easily accessible for a design team, make it worth its time to use, and develop the methodology to be general enough to cover a range of products which varies a lot in their design.

We achieved this by first examining two existing models (STRIDE and OCTAVE) with a similar aim and additionally explore some theory about data security in general, where we focused on how attack and defence usually is carried out.

The next part of the report is the documentation on the analysis and design of our guide. In this part, we identified how to communicate the content of our guide in the best possible way using field specific terminology and figures.
We looked on how to successfully determine the security goals of the system for a system engineer (or similar knowledgeable person) using well-adjusted questions, the best possible setup and known ways to model the security and risk of a system.
The guide itself is chosen to be built up over five individual steps. Each step helps the analyser model the product and find the weaknesses. The first three steps is a tool to model the given product objectively. The last two steps builds a risk analysis and prioritization on where the security has to be improved.

We have chosen two products to evaluate under the model in order to exemplify the methodology for the analyser and to evaluate the guide: A bluetooth door lock system and a Smart TV from Samsung. The door lock system was evaluated using the guide and then tested with a bluetooth sniffing device (Ubertooth). The Smart TV was evaluated only using the guide. We also present a couple of possible attacks on the Smart TV, but they are only theoretical.

The model has been successfully implemented, but some concerns were discovered on the way. In the last part we therefore suggest possible modifications to the method e.g. the way we have modelled the system as a whole, which may have been neglecting some issues, the way we guide the user to implement countermeasures, where we rely on some knowledge of the analyser of what security is and also that the score, by which we grade the aspects of security in the methodology, is not sufficient.
Overall the project is however successful, as we have developed a complete guide with examples, containing a good introduction, a precise guide on the methodology and a well-formed model, which is easily usable and relevant for a manufacturer of a product in the target group.

Indholdsfortegnelse

1 Indledning

Forfatter: Fælles

Dette projekt har til formål at udarbejde en metodisk vejledning, som sætter en produktejer (produktudvikler/-ingeniør/-designer) i stand til på systematisk vis at evaluere sikkerheden i et moderne, forbrugerorienteret produkt, hvori udvidede funktioner (f.eks. nyere trådløse teknologier) er en væsentlig del af produktets feature-sæt, men ligger ud over produktets primære funktion.
Desuden skal projektet evaluere denne vejledning ift. to konkrete produkter, som kan demonstrere anvendelsen og applikationen af vejledningen i "den virkelige verden".

I de senere år har man set et kraftigt spring i mængden af mere avanceret elektronik rettet med internetforbindelse mod den almindelige forbruger. Tendensen er en del af "Internet of Things"[1] og har potentiale til at skabe et intelligent net af apparater overalt omkrings os – både Personal-, Local-, Metropolitan- og Wide Area Networks, som kan ændre vores hverdag totalt. De fleste danskere bærer i dag rundt på en smartphone og det er efterhånden svært at få fat i et TV, der ikke er *smart*. Tendensen har også bredt sig til almindelig forbrugerelektronik som køleskabe, støvsugere, kaffemaskiner og meget andet. Et utal af funktioner bliver implementeret, for at det givne produkt træder ud af mængden.
Konkret betyder det lige nu, at flere og flere apparater kan tilgå internettet i en eller anden forstand. Samtidig vil producenterne gerne have os til at købe nyt udstyr hele tiden og der bruges derfor mange krafter på udvikling. Det betyder også at vedligeholdelse og opdatering af "gammelt" udstyr bliver nedprioriteret[2], hvilket åbner nye muligheder for hackere[3].

Udviklingen kan imidlertid skabe en uoverskuelig stor liste af features, som end ikke ingeniørerne bag, har det fulde overblik over!
Det åbner op for at "uvedkommende" kan få adgang til fortrolig data, at foretage uhensigtsmæssige ændringer i produktet eller at afsløre visse aspekter af produktet, dets funktioner eller dets brugere, som må kunne betragtes som forretningshemmeligheder eller personfølsomme data.

I dag findes der et mindre antal metoder til at modellere trusler i software eller indenfor virksomheden som samlet enhed, men (for så vidt vides) ikke for den samlede hardware-/softwarekombination.
Vores metodik skal tjene til at støtte en udviklingsafdeling i at foretage en fyldestgørende sik-

[1] http://en.wikipedia.org/wiki/Internet_of_Things

[2] Yderligere læsning: http://arstechnica.com/gadgets/2014/01/smart-tvs-smart-fridges-smart-washing-machines-disaster-waiting-to-happen

[3] En hacker er en person, som forsøger at bruge et produkt på en anden måde end det var meningen

kerhedsevaluering af et stykke avanceret forbrugerelektronik med udgangspunkt i ovenstående.

Det er vanskeligt at udvikle en metode til at undersøge et arbitrært produkt, idet man på den ene side skal sørge for at gøre den tilstrækkelig specifik, således at de delelementer man havde forestillet sig, bliver undersøgt, men samtidig skal holde den generel i en sådan grad, at sikkerhedsrisici ikke overses. Vejledningen har følgeligt også en vis grad af fortolkning overladt til producenten, for hvem kender systemet bedre end designeren bag? Vi supplerer kun med et værktøj til at udføre den metodisk og objektivt.
Metodikken til at undersøge produktet er inspireret af den viden, vi har opnået gennem de seneste tre års undervisning og inddrager elementer af især datasikkerhed og software engineering (modellering). Den er udviklet gennem en iterativ proces, hvor vi løbende har evalueret på hvor godt det passer på en række repræsentativt udvalgte produkter, udtaget af den produktportefølje, som vi retter metoden mod.
Rapporten er bygget op med teori, analyse, design, implementering og evaluering af vejledningen først. Selve vejledningen ligger i første del af appendixet og præsenteres i fem velbeskrevne trin med en motivation for de enkelte trin samt en modellering, eksemplificeret ved to repræsentative produkter (en bluetooth-styret dørlås og et Smart TV), som understøtter evalueringen og giver et letforståeligt billede af resultaterne.

2 Problemformulering

Forfatter: Fælles

Vi vil i løbet af projektet udarbejde en konkret vejledning til en generel sikkerhedsmodellering af den type forbrugerelektronik, som er beskrevet i afsnit 1. Vejledningen og dens metodik skal kunne bruges som værktøj til en generel sikkerhedsanalyse, der kan bruges til at afsløre eventuelle sårbarheder i et produkt, præsentere dem på en overskuelig måde og give producenten et godt udgangspunkt for at rette op på dem.

2.1 Mål

Målet for projektet er at udarbejde en abstrakt metode til modellering af sikkerhedstrusler, som kan bruges over en bred kam af moderne former for forbrugerelektronik jf. indledningen, men som dog er så konkret, at den kan bruges uden en dybdegående viden om sikkerhedsbegreber, -trusler og kryptologi.
Vejledningen skal evalueres med en modellering af et Samsung Smart TV (teoretisk evaluering) og et AccessZone-dørlåsesystem fra MVC-Data (praktisk evaluering). Målet med evalueringen er ikke nødvendigvis at finde sikkerhedshuller i de pågældende produkter, men at vurdere brugbarheden af vejledningens metodik i forhold til virkelighedens realiteter og det miljø produktet

må antages at blive brugt i.

2.2 Succeskriterier

- En for virksomheder brugbar, generel vejledning inkl. en modellering, som skaber et godt overblik over sikkerheden i et stykke moderne forbrugerelektronik. Vejledningen skal:
  - ⋆ Identificere angrebsflader for et givent produkt
  - ⋆ Prioritere de fundne sårbarheder efter deres samlede risiko
  - ⋆ Generelt identificere trusler rettet mod disse angrebsflader
  - ⋆ Give vejledning til at øge sikkerheden i et givent produkt
  - ⋆ Beskrive udførslen af sådanne undersøgelser
  - ⋆ Være præcis og kort, men uden at forudsætte stor viden om datasikkerhed hos anvenderen
  - ⋆ Udmøntes i et anvendeligt resultat, som kan bruges i praksis
- Applikation af vejledningen på et par forskellige produkter
- Evaluering af vejledningen ved sammenligning med allerede kendte sikkerhedsproblemer i produkterne og andre antagede trusler

2.3 Arbejdsfordeling

Vi har gennem hele rapporten markeret (straks under overskrifterne), hvem der hovedsageligt har skrevet den efterfølgende tekst. Navnet som står under en overskrift, gælder kontinuerligt for alle efterfølgende afsnit, indtil en ny forfatter står noteret.
Vi har brugt tre forfattere: "Daniel", "Rasmus" og "Fælles".

3 Teori

Forfatter: Rasmus

I dette afsnit lægges et grundlag for produktet af dette projekt, idet vi undersøger hvad der kendetegner et sikkert produkt og, ved at se på de metoder angribere bruger, hvad der skal beskyttes imod. Desuden undersøger vi opbygningen af eksisterende sikkerhedsmodeller, for at lade os inspirere af disse.

3.1 Forsvar

"Sikkerhed" er en meget bred term, som bruges på mange måder. Indenfor IT er det imidlertid kogt ned til tre grundlæggende principper [21] for software, kaldet *the CIA properties*:

Confidentiality Princippet om, at vi altid er garanteret fuld fortrolighed omkring data (læsning, skrivning, eksekvering og måske endda eksistens) – både i transit og i opbevaring (afhængigt af hvilke systemer man kigger på).

Integrity Princippet om, at data eller systemer kun kan ændres af de rette personer og kun på den rette måde.

Availability Princippet om, at data eller systemer er tilgængelige når der behov herfor og performer acceptabelt.

Desuden kan vi udvide begrebet med tre yderligere egenskaber [11], kaldet AAA[4]:

Authentication Princippet om, at en brugers identitet kan bestemmes sikkert og korrekt (eller evt. tillade anonyme brugere).

Authorization Princippet om, at denne identitet er fuldstændig bestemmende for hvilken adgang og rettigheder brugeren har.

Non-repudiation Princippet om, at en handling er fuldstændig ufravigelig, sådan, at det altid kan bevises, hvis en identitet/bruger har udført en given handling.

Disse tre supplerende principper gælder for de værktøjer, der kører på systemet for at sikre de tre oprindelige CIA-egenskaber. De er alle tre nødvendige egenskaber for de adgangs- og kontrolsystemer, som omgiver et sikkert system. De to første handler om at en identitet tildeles en bruger og at denne identitet er konsistent og håndhæves i alle dele af systemet, mens det tredje princip om ufravigeligehed, bedst kan ses som et værktøj til at dokumentere hændelser, *hvis* de tre grundlæggende principper alligevel fejler eller en (i øvrigt tilladt) hændelse skal undersøges[5]. I [18, 25] nævnes også *Accountability* som en kriterie, men det er efter vores mening

[4] http://en.wikipedia.org/wiki/AAA_protocol
[5] Jf. "Se & Hør-sagen"

indeholdt i *non-repudiation* i den definition, vi bruger herover. Målet med de nævnte principper er total sikkerhed.

Indenfor hardware arbejder man ikke særskilt med et sæt kriterier omkring sikkerheden på samme måde som indenfor software. Vi tror ikke, at der tidligere har været særligt stort fokus på sikkerheden som særskilt disciplin, så da softwaren har gjort sit indtog i hardwaren, er de samme principper blevet dækkende for hardware-/softwarekombinationen også; dette er dog med den naturlige, fysiske begrænsning hardwaren giver. I [18] bruges *AAA* som sikkerhedsprincip og her anføres det, at for den fysiske hardware, er det kun et spørgsmål om sikring af den fysiske adgang. Som vi ser på i sektion 3.2, er det dog lettere sagt end gjort!

Ovenstående skal bruges i det senere arbejde med metoden, for med udgangspunkt i disse seks principper, kan vi guide producenten til en korrekt forståelse af, hvad der kan kendetegnes som sikkerhedsbrister i vedkommendes system. Det er nødvendigt, at vedkommende guides på et generelt plan, idet vi umuligt kan designe en metode, som fuldstændigt dækker de mange, alsidige produkter, som vi har i målgruppen.

3.2 Angreb

Udover den defensive del er det relevant, at have en vis viden om, hvordan en angriber finder sårbarheder i et givent system.

Denne info er heldigvis ikke så skjult, fordi mange efterhånden ernærer sig ved at udføre *penetration tests*[6] imod diverse systemer og branchen generelt fungerer med en *open source*[7]-tilgang ift. både teknikker og software. En pen-testers[8] fremgangsmåde er meget lig den for en *black hat*-hacker[9]. Figur 1 viser et typisk eksempel på det samlede forløb af en pen-testers arbejde. Vi er især interesserede i at se på trinnene hvor sårbarhederne identificeres og udnyttes, fordi det er den slags værktøjers fremgangsmåde, vi potentielt skal hindre. Dermed ikke sagt, at er beskyttede, men mængden af automatiserede værktøjer, som scanner virksomheder og deres netværk hver dag, er ganske overvældende, så det kan reducere frekvensen hvorved (forsøg på) angreb sker, betragteligt.
Groft kan selve de to relevante trin opdeles på følgende måde [9]:

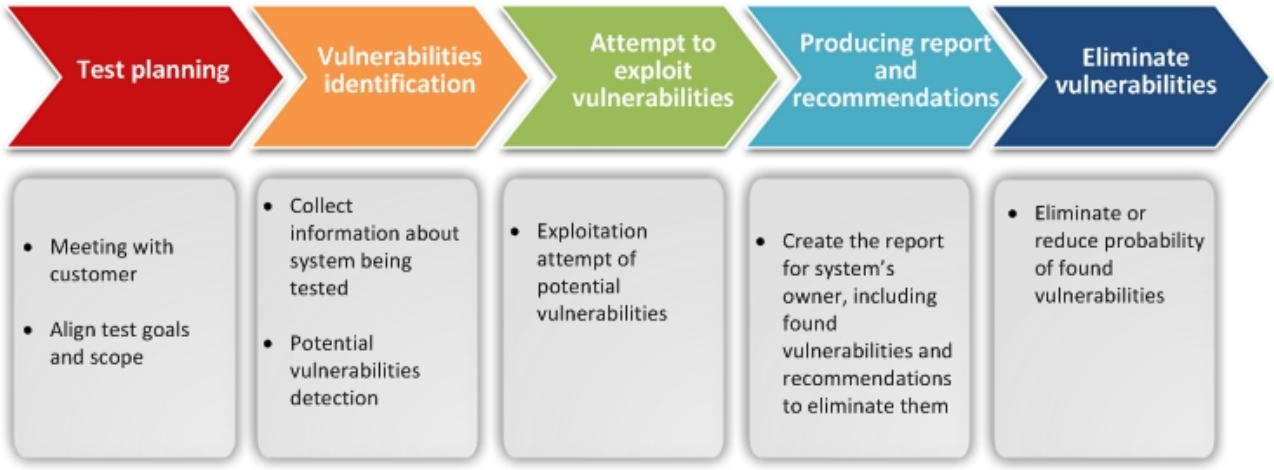
[6] Aftalte angreb på systemer eller virksomheder mhp. at af dække de reelle sårbarheder der findes her og efterfølgende rapportere dem til systemejeren

[7] Princippet om, at kode er frit tilgængelig og frit kan redistribueres under visse betingelser; http://en.wikipedia.org/wiki/Open_source

[8] *Penetration tester* – en person som udfører penetration tests

[9] En ondsindet hacker. Arbejder typisk ud fra et egoistisk, ideologisk eller økonomisk grundlag

[10] Kilde: http://auditagency.com.ua/images/pentest1_en.jpg



Figur 1: Den typiske arbejdsproces for en pentester[10]

1. Indhentning af relevant information gennem fysiske besøg, snak med relevante parter, profilering af ansatte via f.eks. sociale netværk, IP adresser, enheder osv. (også kendt som *social engineering*[11]).

2. Scanning for sårbarheder med automatiske værktøjer (aktivt eller passivt, afhængigt af hvor tydelig angriber vil være) eller ved fysisk gennemgang af produktet (indkøb af en model) mhp. at undersøge komponenter eller udføre analyse af koden (reverse engineering[12])

3. Validering af sårbarheder manuelt (se i sårbarhedsdatabaser[13] og kombinér med oplysninger fra trin 1)

4. Udnyttelse af sårbarheder

Specielt interessant er trin 1, for det er ensbetydende med, at selvom man indenfor vores målgruppe af produkter kan have at gøre med specialiserede og obskure hardware-/softwarekombinationer, så bør man forvente, at en angriber forbereder og forsøger at skaffe denne information. Det kan bl.a. ske gennem gennemlæsning af tekniske manualer, social engineering eller direkte adgang til fabriksområder.

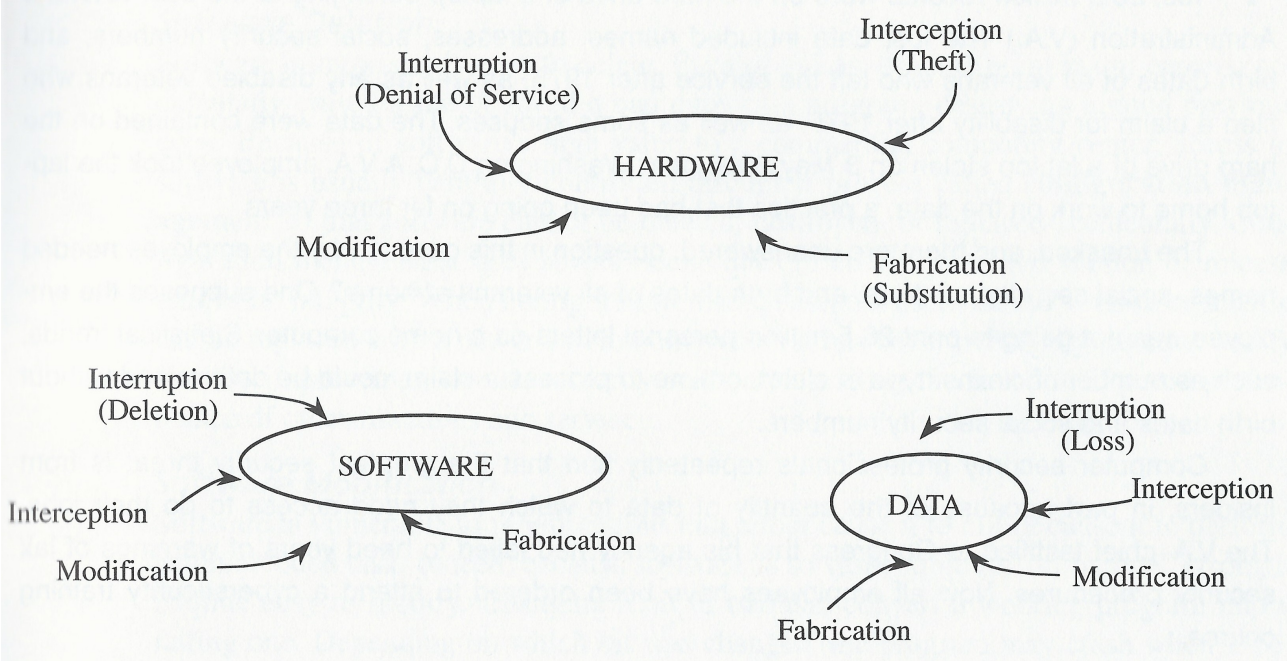
Vores metode skal som sagt rette sig mod en samlet hardware-/softwarekombination, imens ovenstående, både angreb og forsvar, er møntet på softwaressystemer. Mange af disse produk-

[11] Se forklaring længere nede i dette afsnit

[12] Reverse engineering er processen, hvori man undersøger struktur, funktion og operation af et stykke kode eller et produkt, for derved at kunne opnå forøget viden om produktet eller dets teknologiske principper; http://en.wikipedia.org/wiki/Reverse_engineering

[13] Databaser over sårbarheder vedligeholdt af relevante myndigheder eller organisationer. http://web.nvd.nist.gov/view/vuln/search og https://cve.mitre.org/cve/ er meget udbredte og indeholder også henvisninger til løsninger

ter består dog af sammenlignelige elementer (f.eks. webservere), for hvilke sårbarhedsanalysen vil foregå på præcis samme måde. Derfor kan henvisninger til disse værktøjer gøre nytte for en producent, skulle denne ønske at gå i dybden udvalgte elementer fra resultaterne af metoden.
Den generelle mængde angreb kan f.eks. kategoriseres som på figur 2. Den samme opdeling vil



Figur 2: Sårbarheder i et computersystem. Kilde: [21]

formentlig kunne finde anvendelse til at skabe et overblik eller kategorisering af sårbarheder i vores vejledning. Hardwarens sårbarheder, som man kan have tendens til at overse, hvis man er mere softwareorienteret, er her inkluderet. Det ses, at det reelt er ens kategorier for alle tre undersystemer.
Vi kan ikke påstå, at denne segmentering er endelig, men vi mener, at den er et rigtigt godt udgangspunkt. En udvidelse kan ske i form af de de sikkerhedskriterier, som identificeres i tabel 1.
En vigtig ting som ikke er medtaget på ovenstående figur, men som er en essentiel del af en pen-testers metoder, er *social engineering* [1]. Her handler det om at fremskaffe sin information ved at bedrage de personer, som måtte have relevans for det aktuelle emne, som undersøges. Det kan f.eks. være at optræde som en underleverandør af firmaets netværkssupport og udbede sig et password, henvende sig direkte til ledende medarbejdere under dække af at være en relevant forbindelse eller slet og ret at se "vigtig" ud (sikkerhedsvest og beskyttelseshjelm!), spankulere ind og hente et par harddiske fra rackskabet.
Dette er ganske reelle trusler[14], som nemt kan blive overset blandt de mere "højteknologiske" angreb.

[14]https://www.us-cert.gov/ncas/tips/ST04-014

En anden til tider negligeret trussel, er firmaets egne medarbejdere. De kan gennem deres tid på arbejdspladsen have opnået en dyb viden om sikkerhedspraksisser (inkl. legitimationsoplysninger), forretningshemmeligheder o.lign. Årsagerne til, at man pludselig vender sit firma ryggen kan være mange: En uretfærdig afskedigelse, en ekstern økonomisk klemme, uvenskab eller slet og ret skødesløshed, som når man mister en bærbar computer på en forretningsrejse. Dette kan udnyttes målrettet af en angriber, for ligesom ved social engineering, har vi her med irrationelt handlende mennesker at gøre, som kan manipuleres og snydes for at udtrække data fra virksomheden.

Med ovenstående er det dog også vigtigt at bemærke, at det mønter sig på enkeltstående dele af et system, men når man kombinerer disse til en samlet enhed, kan andre sårbarheder opstå. Det samme gælder for selve succeskriteriet: At systemet er sikkert. Det er bare ikke muligt at bevise dette!
Selvom en given service er sikker ift. en given angrebstype og vi kan argumentere herfor på baggrund af scanninger og kendte sikkerheder, så kan man ikke garantere, at andre sårbarheder ikke eksisterer alligevel og især ikke, at produktet, når det hele kobles sammen, kan opfylde de samme kriterier i kommunikationen imellem funktionerne [11]. Dette kan faktisk direkte invalidere antagelser som er gjort omkring disse funktioner (som f.eks. at data er korrekt formateret). Det eneste som rent faktisk kan siges, er, at hvis en given funktion eller komponent i det samlede system er sårbart, så gælder dette også det samlede system og således kan en undersøgelse af sårbarheder pr. delkomponent give mening for os og producenten.

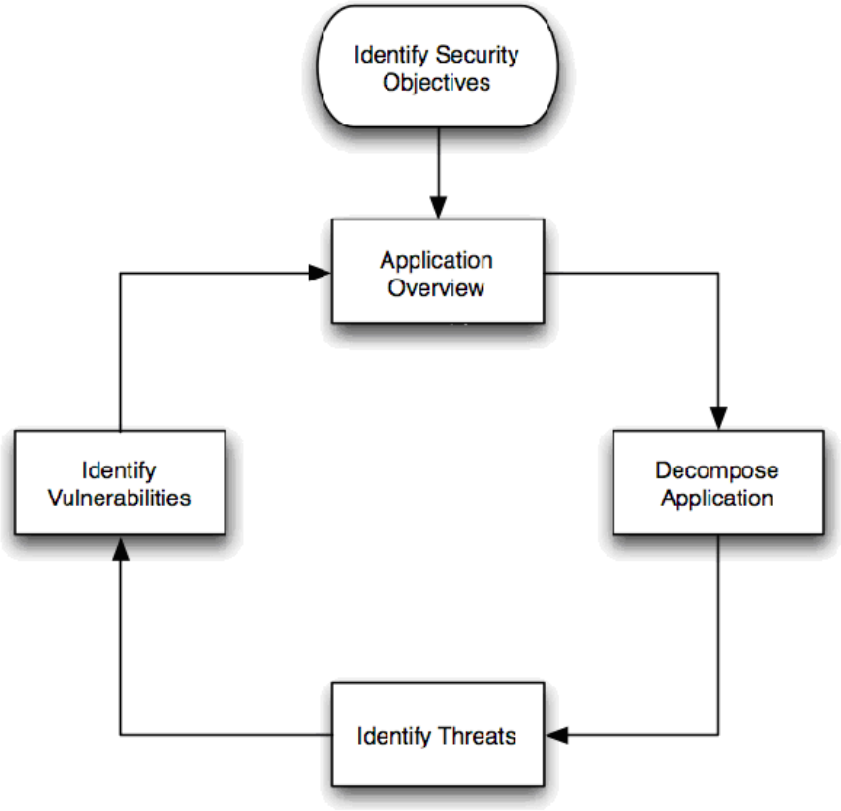
3.3 Eksisterende modeller for sikkerhedsevalueringer

Modelleringer er vidt udbredt, når man skal undersøge eller bygge nye systemer. Et ideelt produkt udvikles vha. f.eks. automatons[15], som kan bevise den korrekte eksekvering af programmet ift. både kørsel og resultat. Det samme kendes indenfor datasikkerhed, hvor man f.eks. kan undersøge en sikkerhedsprotokol vha. `AnB`-notationen, som er en idealiseret fremstilling af de forskellige dele (interessenter, nonces, krypterede beskeder mv.), som kan verificeres i et program.

En generel trusselsmodelleringsproces finder vi f.eks. hos OWASP[16]. Denne process bygger på Microsofts' trusselsmodelleringsproces, som konkret udmønter sig i STRIDE-modellen, som vi kigger på senere.

[15] En såkaldt state machine, er en graf der f.eks. bruges til at modellere eksekveringen programmer på en computer

[16] The Open Web Application Security Project – https://www.owasp.org/index.php/About_OWASP



Figur 3: Trusselsmodelleringsprocessens fem trin iflg. OWASP og deres indbyrdes flow. Kilde: [23]

Denne generelle model beskrives som en gentagen proces, hvor man i første omgang skal definere successkriterierne for sikkerheden i systemet og siden repetitivt gennemføre processen for at skabe et overblik, udspecificere systemet i subsystemer og identificere og behandle trusler her. Det ses, at tilgangen med at opdele produktet i mindre dele også bruges her, på trods af de risici det kan medføre, hvis man herved overser trusler, som udspringer af undersystemernes sammenkobling (som beskrevet i afsnit 3.2).

Reelle modeller, der som output direkte viser trusler imod det undersøgte produkt, findes indenfor software f.eks. i form af STRIDE[17] [11] eller indenfor virksomhedsorganisationer i form af OCTAVE[18] [20]. STRIDE er det konkrete produkt af Microsofts trusselsmodelleringsproces, som anbefales af OWASP [23].
De er begge mere "business"-orienterede, så de er nemmere for ikke-tekniske personer at bruge. Fokuset er på selve identifikationen af trusler, mens identifikationen af aktiver i STRIDE

[17]STRIDE®-modellen ("**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service & **E**levation of privilege")

[18]OCTAVE®-modellen ("**O**perationally **C**ritical **T**hreat, **A**sset, and **V**ulnerability **E**valuation"). Vi tænker både på den ældre OCTAVE, OCTAVE-S og OCTAVE Allegro, idet der i opbygningen ikke er væsentlige forskelle

tillægges en lidt mindre betydning; for OCTAVE's vedkommende er det dog ikke tilfældet.

De to modeller ligner hinanden relativ meget i opbygningen; de er dog lidt forskellige i deres konkrete indhold qua de forskellige målgrupper, men de består grundlæggende af tre trin:

Organisatorisk evaluering Opbyg en model/profil af det, der ønskes undersøgt

Teknologisk evaluering Analysér aktiver og identificér trusler

Strategiudvikling Lav en risikoprofil af truslerne og implementér en løsning

Først opbygges en model af systemet/produktet for at få skabt et godt, destilleret overblik og opdele det store, samlede problem (systemet/produktet) i mindre delproblemer [11]. Der er især fokus på at afdække de nuværende forhold, med inddragelse af ressourcepersoner, for at få et præcist billede af de forskellige "information containers" værdi (ud fra en subjektiv vurdering af ressourcepersonerne) og de sikkerhedsforanstaltninger, som pt. er i brug [20].

Dernæst kigger man på hvilke komponenter, der er essentielle for sikkerheden. Dette er i begge modeller en vurdering, som foretages af den ansvarlige person for evalueringen. I OCTAVE antager man, at den ansvarlige person har tilstrækkelig viden til at foretage disse undersøgelser[19]. I STRIDE gøres det ud fra de seks nøgleord, for hvilke STRIDE er et akronym: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service & **E**levation of privilege. Disse egenskaber er direkte linket til de i sektion 3.1 nævnte sikkerhedskriterier som illustreret i tabel 1.

Security properties	**Threats**
Confidentiality	Information disclosure
Integrity	Tampering
Availability	Denial of service
Authentication	Spoofing
Authorization	Elevation of privilege
Non-repudiation	Repudiation

Tabel 1: Sammenhængen mellem "standard"-sikkerhedskriterierne og trusler som gengivet i [11]

Analysen består af at holde disse seks trusler op imod den mængde underproblemer, som er blevet genereret vha. modellen i det forrige trin. Derved får de(n) ansvarlige et bedre og mere præcist udgangspunkt for analysen, fordi vedkommende får en konkret vejledning i hvad der

[19]Det står der ikke direkte, men der står, at "*The analysis team identifies (. . .) then determines the extent to which (. . .) [a] component is resistent to (. . .) attacks and establishes the technological vulnerabilities*", hvilket forstås som, at de altså selv forestår undersøgelserne

skal kigges efter. Man får direkte at vide hvad der skal undersøges for, så sandsynligheden for, at det er mere dækkende, er større, end hvis hver eneste organisation som skal sidde med dette fremover, skal opfinde den dybe tallerken og nedfælde en række generelle tilfælde hver gang. Risikoen er herved, at de netop ikke er tilstrækkelig dækkende og nogle aspekter bliver overset. Jf. ovenstående, er STRIDE mest møntet på de traditionelle trusler, mens OCTAVE også *kan* dække de fysiske trusler (*social engineering* og *insider threats*, se 3.2).

3.3.1 Tildeling af score

I dette trin, og inden man kigger på at imødekomme problemerne, bruger man ofte en skala for at klassificere truslerne [15]. Den skala vi har fået demonstreret i undervisningen, bestemmer risikoen ud fra produktet af to scores baseret på sandsynligheden for, at en given sårbarhed udnyttes og denne sårbarheds konsekvens for systemet. Skalaen herfor er arbitrær og egentlig underordnet, men det er fremmende for brugen, hvis man kan skelne de forskellige niveauer fra hinanden [5]. Vi ser nærmere på et sådant system i 4.2.4, idet det ikke indgår i disse modeller. Tidligere benyttede man systemet DREAD [5], hvor brugeren guides til at foretage klassificeringen pba. specifikke, isolerede egenskaber i modsætning til bare at bede om at få vurderet "konsekvensen" af udnyttelsen. I [15] eksemplificeres med seks forskellige spørgsmål:

Damage potential Hvad er konsekvensen, hvis sårbarheden udnyttes (hvor meget data afsløres)?

Reproducibility Hvor besværligt er det at gentage angrebet?

Exploitability Hvor meget viden kræver det at udføre angrebet?

Affected users Hvor mange brugere er omfattet af sårbarheden?

Discoverability Hvor nemt er det at opdage sårbarheden?

Reputation I hvor høj grad er virksomhedens renomé/kundetillid udsat?

Det sidste spørgsmål, er et eksempel på, hvordan man yderligere kan tilpasse spørgsmålene til sit behov.
Tanken er så, at man tillægger en score fra f.eks. 0-5 for hvert spørgsmål ift. truslen og endeligt udregner en gennemsnitsscore pba. dette (se tabel 2).

Metoden er god, fordi den kan udvides efter behov og man samtidig kan forfine scoren mere præcist end ved blot at gange sandsynlighed og konsekvens. Den kan dog være for subjektiv i og med den større skala giver analytikeren mulighed for generelt at holde tallene i den høje ende, hvis han er skeptisk (i modsætning til en skala på 1-3, hvor hvert tal kan karakteriseres

Sårbarhed	**D**	**R**	**E**	**A**	**D**	I alt
SQL-injection	5	3	2	5	1	$\frac{5+3+2+5+1}{5} = 3,2$
Packet injection på produktionsnetværk	1	1	1	5	5	$\frac{1+1+1+5+5}{5} = 2,6$

Tabel 2: Eksempel på brug af DREAD til at klassificere trusler

præcist) og fordi den er svær at vurdere ift. en skala med kun to-tre spørgsmål.
Samtidig prøver man at lave et samlet billede ud fra nogle tal, som slet ikke har noget med hinanden at gøre, når man ganger dem allesammen sammen og tager et gennemsnit [5].

3.3.2 Anbefaling af sikkerhedsforanstaltninger

Endeligt skal der udvikles og implementeres løsninger. I STRIDE følger løsningerne til dels af de seks sikkerhedskriterer (tabel 1), hvor man ser på de tilgængelige teknologier og udvælger relevante. Det anbefales desuden at overveje, hvorvidt teknologien kan gøre en positiv forskel og reelt vil blive brugt i scenariet [11]. Der fordres dog en vis teknologisk indsigt, idet der ikke gives en nærmere uddybning af hvilke typer "sikkerhed" der kan afhjælpe de givne trusler.
I OCTAVE er informationen om løsninger ligeledes sparsom, så der påhviler her det analyserende team en endnu større byrde hvad angår datasikkerhedsteknisk viden.
For at hjælpe i løsningsprocessen, foreslås det i [11], at man opbygger *threat trees*[20] og ud fra disse prioriterer og afhjælper de identificerede problemer. Der tages forbehold for, at opbygningen af disse er meget kompliceret, så for ikke at skulle "opfinde den dybe tallerken" hver gang, bruger man hos Microsoft *prebuilt threat trees*, som løbende udbygges.
For at hjælpe med at identificere endnu ukendte trusler, kommer *attack patterns*[21] i spil. Som beskrevet på Wikipedia, så er "angrebsmønstre" en måde at kategorisere sammenlignelige angreb (på samme sårbarhed, men som dog kan variere i f.eks. specifik udførsel eller payload), som kan kopieres til at lave generelle beskrivelser for de forskellige dele af modellen. Det ser vi nærmere på i afsnit 4.2.5.

Én ting, som er medtaget i OWASP's generelle process (figur 3), men ikke fremgår som en del af STRIDE og OCTAVE, er processen med at identificere sikkerhedsmålene/succeskriterierne for systemet. Heri er indeholdt overvejelser, som både kan motivere brugen af modellen og hjælpe til at bestemme risiciniveauerne bedre (så de afspejler realiteterne bedst muligt). Dette er f.eks. overvejelser om, i hvor høj grad brugerdata skal beskyttes og hvor vigtig tilgængeligheden af en given service er.

Begge modeller lægger op til, at man bruger dem på en dynamisk måde, hvor team'et er klar til

[20] Sæt reference til afsnittet
[21] http://en.wikipedia.org/wiki/Attack_patterns

at gå baglæns, hvis ny viden bliver tilgængelig undervejs. Hovedformålet med dette er især den konstante ændring i forholdene omkring systemet, for hvilket analysen jo gerne skal afspejle den aktuelle situation. Det svarer i øvrigt til hvordan man også indenfor risikoanalyse generelt håndterer dette [4, 6] (se også figur 5 i sektion 4.2).

4 Analyse

Forfatter: Rasmus

I dette afsnit kigger vi på hvordan vi opnår målet og succeskriterierne som de er formuleret i afsnit 2. Vi vil afdække de udfordringer som er relevante for metodens (og dermed projektets) umiddelbare succes og se på hvordan de kan håndteres (dette kan være f.eks. i form af modtageren eller miljøet, som metoden skal bruges i).

Målet for projektet er at opbygge en anvendelig metode og skrive en vejledning med udgangspunkt i denne. Anvendeligheden afhænger imidlertid først og fremmest af modtageren, for hvem budskabet skal give mening og passe til dennes vidensniveau. Endvidere skal metoden, for at være relevant, have en gennemskuelig, anvendelig og logisk opbygning og endeligt synes besværet værd for både ledelse og udviklingsafdeling, således, at der investeres den nødvendige tid i de relevante undersøgelser. Det er ligeledes vigtigt, at vi tager højde for den virkelighed som omgiver produktet, hvilket igen bidrager til, at metoden kan skabe værdi for producenten. Endeligt er det et ultimativt (funktionelt) krav, at den reelt kan hjælpe til at afdække sikkerhedsrisici ved produktet.

4.1 Effektiv kommunikation

At kommunikation af vores metode og vejledningen hertil er effektiv, er helt essentielt for projektet! I dette afsnit ser vi kort på kommunikationsteori og hvordan det kan bruges i vores vejledning.
Grundlæggende består kommunikation af fem enheder [14], som er relevante for os at overveje i forhold til udformningen af vores vejledning:

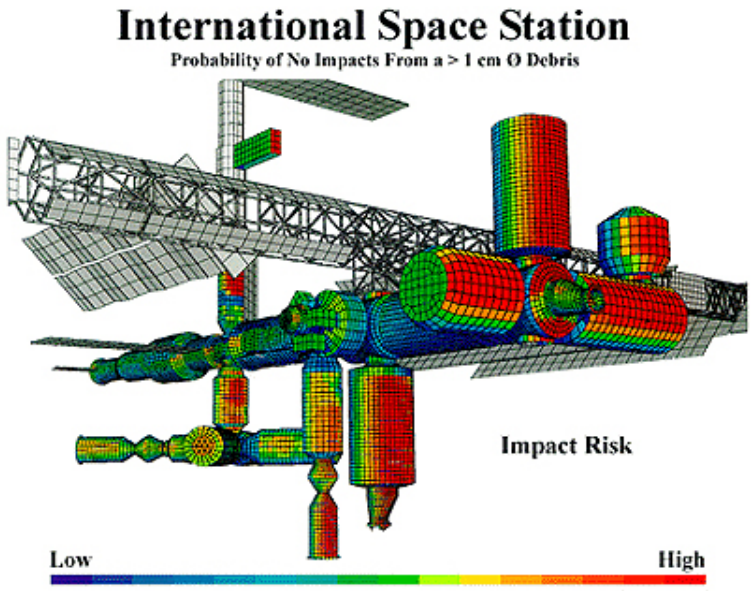
- Afsender
- Budskab
- Medium
- Modtager
- Effekt

Som **afsender** er det vigtigt at fremstå relevant, vidende og seriøst overfor **modtageren**. Hvis ikke modtager forstår budskabet, er det afsender, der har fejlet. Dette sikres ved først og fremmest helt subtilt ved at tilpasse sproget i vejledningen til noget mere "brugsanvisningsagtigt" i modsætning til en teknisk rapport (som denne). Forskellen ligger især i formen, hvor det i rapporten er beskrivende og forklarende og har antydning af at være rettet mod en ligemand, skal det i vejledningen være i bydeform, idet der skal gives direkte instrukser om en fremgangsmåde. Ydermere kan vi sikre relevansen ved, at budskabet tager udgangspunkt i producentens virkelighed og er mærkbart rettet mod denne. Det er vores intention, at modtageren er en udvikler med "fingrene godt nede i produktet" og med en baggrund som f.eks. software- eller elektroingeniør og at vedkommende evt. udfører analysen sammen med en person, som har et kendskab til de typiske angrebsmønstre der ses mod software (over en bred kam) i dag.

Vores **budskab** i vejledningen kan fremhæves ved at sikre, at metoden er bygget op på en logisk måde og at vejledningen følger en genkendelig opsætning, så den virker logisk for modtageren og vedkommende desuden genkender de metoder, som er brugt. Det første kan vi sikre ved at bruge en opbygning, som den kendes fra mange andre vejledninger, risikoanalyse[22], gymnasiet[23] og meget andet, hvor indholdet præsenteres i en stigende kompleksitet og dybde startende fra simpel identifikation af elementer og til sidst en vurdering af truslerne. At vejledningen virker genkendelig, kan desuden opnås ved, at den har træk som andre modeller for sikkerhedsevalueringer. Her bevæger man sig dog over i analysen af hvilket kvalitativt indhold vejledningen skal have; dette ser vi på i afsnit 4.2 og 4.3.
Ydermere er det vigtigt, at der gives indtryk af en fælles kultur (og dermed identitet) mellem afsender og modtager, fordi det igen faciliterer en mere effektiv overførsel af budskabet til modtageren, hvis vedkommende kan identificere sig med afsenderen og man, rent semantisk (som udgangspunkt), har den samme forståelse af et givent udtryk. Dette kan vi gøre ved at bruge udtryk, som afspejler branchen og ved at tænke over sproget.

Vores fysiske **medie** kan vi ikke ændre på i dette tilfælde, men vi kan præsentere indholdet på en nem og let tilgængelig måde, fordi det hæver brugbarheden af resultaterne væsentligt. Vejledningen kan være nok så god, men hvis den består af 10 tætskrevne sider uden logisk opbygning og giver et output, som bare er ren tekst, er den på grænsen til at være ubrugelig i praksis, fordi man ikke hurtigt kan udlede meningen og afkode et resultat. Derfor skal den ligesom de andre modeller, opbygges i trin.

[22] http://en.wikipedia.org/wiki/Risk_management#Method
[23] De tre taksonomiske niveauer



Figur 4: Et eksempel på et heat map, som hurtigt kan aflæses. Det viser ISS' risiko for sammenstød med objekter $> 1\,cm\,\varnothing$

Et rigtig godt eksempel på hvordan man hurtigt kan afkode data, er *heat maps*, som det i figur 4. Det viser sandsynligheden for sammenstød med objekter på ISS. Havde dataen været fremsat i et excel ark, havde det været langt vanskeligere at identificere de mest udsatte områder af rumstationen og man havde måske tilmed overset vigtige steder.
Den samme tankegang skal overføres til vores vejledning! Resultatet af denne skal gerne have en vis **effekt** og for at opnå denne, skal resultatet fremstå i en brugbar form. Det vil vi gøre ved, sideløbende med vejledningens trinvise gennemgang af metoden, at opbygge en model i trin svarende til metodens — og på samme måde som på figur 4, farves den for at gøre resultatet nemt at afkode.

Modellen er hermed endnu en vej, udover den rene tekst, hvorved vi kan udtrykke budskabet af vores vejledning. Opbygningen af denne model skal indkorporere elementer af genkendelighed fra samme "kultur" som modtagerens, som diskuteret tidligere, for at opnå det bedste resultat. Vi ønsker også med denne vejledning, at der gives et brugbart output, som kan resultere i en ændring (*effekt*) overfor producenten og dermed en øget værdi for denne. I kraft af dette, er det derfor vigtigt, at når vi skriver vejledningen, har fokus på, at den er rettet direkte mod en praktisk applikation på et produkt. Selvom det kun er en arbitrær og generel vejledning, skal der i videst muligt omfang, skabes sammenhæng med virkeligheden og de trusler produktet her udsættes for.
Årsagen hertil er specifikt, at man ofte i virksomheder har fokus på bundlinjen, hvorfor sikkerhedsteamet kan have noget svært ved at gøre deres sag gældende [13]. Samtidig kan ansvaret være svært af fastlægge eksakt, men det er dog udenfor vores opgaves område.

4.2 Indhold

Forfatter: Fælles

Metodens udformning og dermed indholdet af vejledningen, er selvfølgelig det vigtigste element i dette projekt. Meget af den viden som vi vil inddrage i dette, er akkumuleret over tre år, men vi skal her prøve at pege på væsentlige elementer, som vil være med til at give vejledningen sin form.

Ved at basere vejledningen direkte på principper fra datasikkerhedsteorien opnår og indkorporerer vi samtidig den genkendelighed og kultur som vi bl.a. har set i andre modeller (3.3), som vi ønsker (jf. 4.1). Således skal vejledningen bygges op over de, indenfor risikoanalyse, basale trin (her baseret på [22], men [23] har en lignende opstilling):

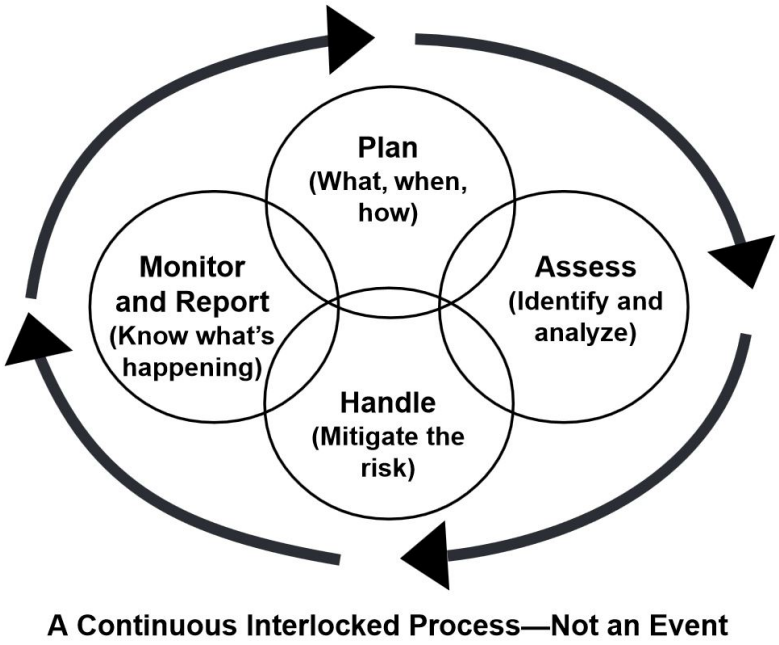
1. Identifikation af aktiver/værdier
2. Identifikation af sårbarheder
3. Vurdering af sandsynligheden for udnyttelse
4. Vurdering af sikkerhedsforanstaltninger
5. Anbefaling af modforanstaltninger

Der findes også andre lignende måder at præsentere ovenstående indhold på, f.eks. catchphrasen *"Identify, Assess, Treat, Mitigate"*[24], men pointen er den samme: Identificér værdier og sårbarheder, analysér, vurdér og håndtér disse, iværksæt foranstaltninger (og eventuelt monitorér og rapportér). Det sidste punkt binder metoden sammen med det første og det er således en cyklus, som kan gentages og løbende give en risikoanalyse af de monitorerede elementer [6]. Et eksempel herpå ses i figur 5.

4.2.1 Sikkerhedsmål

For at kunne identificere sine aktiver i systemet, mener vi, at der kræves, at producenten til en vis grad har gjort sig klart, hvilke sikkerhedsmål vedkommende ønsker at opnå. For at afdække disse, kan man bruge spørgsmålene i [23], som bl.a. fokuserer på værdi- og imagetab ved kompromitering, risikovillighed, relevante love og regler mv. Dette er vigtigt og kan motivere arbejdet, men flere af disse spørgsmål handler om de konkrete elementer i produktets og konsekvensen af en sårbarhed i disse. Fordi vores vejledning skal afdække disse og fjerne antagelser, som måtte være heromkring, så kan vi ikke spørge ind så tidligt i processen.
Imidlertid giver det god mening at det inddrages, når det kommer til vurderingen af de enkelte funktioners konsekvens, fordi der her kan åbnes for nye tankerækker hos analytikerne. Idet vi

[24] http://en.wikipedia.org/wiki/Risk_management#Method



Figur 5: De fire elementer i risikomanagement. Kilde: [6]

er interesserede i at fjerne de subjektive antagelser, som producenten kan have om produktets sikkerhedsegenskaber (eller mangel på samme), er det et stort plus, hvis man kan facilitere dette ved at stille spørgsmålene.

4.2.2 Modellering af systemet

Det vil i første omgang være vigtigt at kunne identificere truslerne korrekt for at have et brugbart værktøj, men for at gøre dette, skal vi være sikre på at kunne dække hele systemet af både hardware og software! Det er præcis på dette punkt, at vi adskiller os fra kendte trusselsmodeller, som beskrevet i afsnit 3.3.
Vi skal vurdere en fasttømret kombination af hardware og software og derfor giver det ikke mening, at de to ting adskilles i vejledningen; en service, som ikke kan nås "udefra" er ligegyldig i denne sammenhæng, for den kan ikke tilgås af (den potentielt fjendtlige) forbruger eller andre, når produktet er udgivet. Vores vejledning skal således kunne modellere denne gensidige afhængighed mellem ydersiden og indersiden, software og hardware!

Det giver bedst mening at modellere systemet som en boks, hvor inputs markerer "åbningerne" ind til produktet, som så bindes sammen med de services, der kan nås herfra. Vi har overvejet, at det vil kræve et vist produktkendskab at kæde disse inputs og services sammen korrekt (for det er essentielt for metoden, at de er det – ellers kan indtil flere sårbarheder forblive skjult), men der må forudsættes, at en tilstrækkelig vidende kapacitet sættes på opgaven med at udføre vejledningen (som der også gøres i introduktionen til STRIDE). Det er endvidere en ekstra

overvejelse værd, at hverken vi eller producenten har mulighed for at sige noget om, hvorvidt den hardwareimplementerede adskillelse af funktionerne (som der må være) er tilstrækkelig robust til, at vi kan lave denne segmentering. Vi forudsætter jo netop med denne direkte sammenkædning af input og deres relaterede services, at det er præcis det dataflow der sker på "indersiden" – og intet andet!

Vi vil gerne inkludere en segmentering af de forskellige inputs her — både fordi det deler opgaven med at identificere input i to mindre bidder, hvilket gør den lettere tilgængelig for brugeren, men også fordi vi tror, at der kan være en sammenhæng mellem deres kategori og hvilke services de er forbundne med.
Vi har overvejet to inddelinger: Kablet/trådløst og producent-/forbrugerinputs. Vi vælger imidlertid at gå videre med opdelingen efter den fysiske måde input'et fungerer på, fordi det bedst vil gå i tråd med opdelingen i "typer", som forhåbentlig letter identifikationen, fordi en producent kan koncentrere sig om alle de trådløse først og så de fysisk forbundne. Opdelingen efter hvilke grupper, som input'et er beregnet til, er et dårligere valg, fordi det lægger nogle antagelser omkring brugen af input'et ind over og det er netop dem, vi er interesseret i (i videst muligt omfang) at undgå at medtage i modelleringen. Det kunne være antagelser om en bestemt brug, der er skyld i at sikkerhedsbrister opstår, fordi man eksempelvis forventer et korrekt formateret input[25]. Vi skal imidlertid, omend ikke invalidere disse antagelser, så holde dem ude af vurderingen og se helt nøgternt på produktet, for dermed at sikre en heldækkende sikkerhedsanalyse.

Vi har desværre ikke tilstrækkelig hardwaremæssig viden til at vide, om en segmentering af inputs og services kan garanteres fysisk med kredsløb, men kan det det, har vi med sammenkædningen af inputs og services, implementeret endnu et stærkt element, som også går igen i de andre modeller og øger genkendeligheden: Data flow diagrammer (DFD)[26] (dog en variation heraf).
Dette kan bruges til at opbygge en model, som inkorporerer de indledende øvelser med identifikation af produktets opsætning og således giver producenten et komplet overblik over inputs, services og deres indbyrdes forbindelser.
Modellen kan bruges som værktøj til at præsentere resultaterne af de undersøgelser, som laves i de efterfølgende trin, hvor en, til en hvis grad subjektiv, vurdering af de forskellige services værdi og kritiske niveau kan indføres og dermed skabe et godt overblik. Dette kan f.eks. gøres ved at bruge risikomatrixerne på en lidt anderledes måde, så man laver skalavurderingen, men så i stedet for at markere med et tal, farvelægger elementerne og derved opnår samme effekt

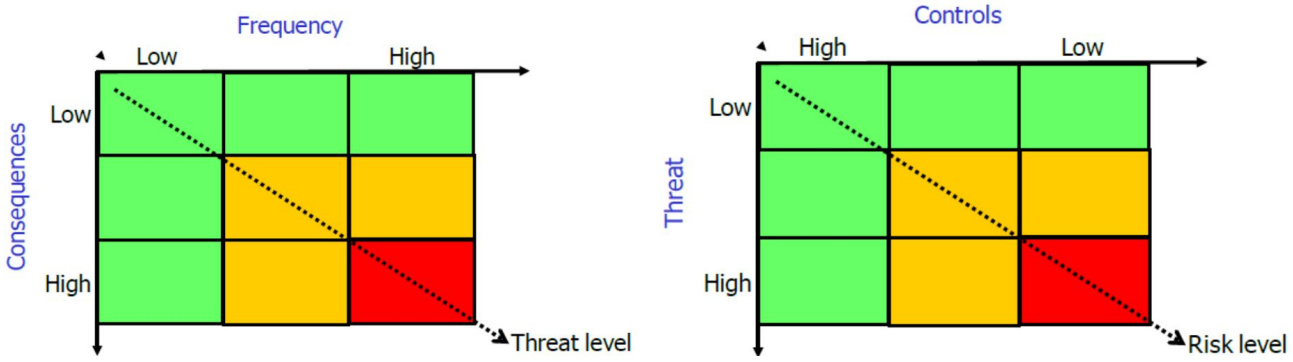
[25] F.eks. injections i databaser, crashes eller buffer overflows
[26] http://en.wikipedia.org/wiki/Data_flow_diagram

som et heat map (se figur 4), som diskuteret i 4.1. For at følge denne opbygning videre, skal vi bruge farvelægningen som gennemgående redskab og også finde en måde at lægge denne vurdering ud på input'et, men hvor selve input'ets sårbarhed også tages med ind over og vi opnår en samlet vurdering som ved trussels- og risikomatrixen (figur 6).

I et efterfølgende trin kunne man udarbejde en oversigt til hvert fundet input, som angiver hvem der er tiltænkt adgang til disse. Derved kan producenten få et overblik over, hvem der har adgang til de forskellige dele af systemet og i det endelige resultat, kan vi bruge dette til at lægge fokus på inputs, som *ikke* er tiltænkt forbrugeren, fordi man kunne antage, at der her i endnu ringere grad vil være truffet de korrekte sikkerhedsforanstaltninger og samtidig, på et eller andet sæt, ligge fokus på "forbrugerens inputs", fordi de, grundet deres tydelighed, ser en højere frekvens af angreb.

4.2.3 Risikomatrixen

Det fjerde trin i vejledningen vil give anledning til at indkorporere et andet anderkendt værktøj fra risikoanalyse: Risikomatrixen [7], som vha. farver faciliterer hurtig idenfikation af de mest kritiske dele. Vi vil benytte en variation af denne, hvor man først plotter truslens konsekvens mod dens frekvens og dernæst plotter dette resultat mod de sikkerhedsforanstaltninger, som eksisterer imod den enkelte sårbarhed/trussel [10]. Et eksempel herpå ses i figur 6.



(a) Trusselsmatrix – hvis en trussels konsekvenser er store og dens hyppighed er høj, vurderes en trussel til at være stor

(b) Risikomatrix – hvis en trussel er stor og sikkerhedsforanstaltninger mod denne er lave, vurderes risikoen til at være stor

Figur 6: Trussels- og risikomatrix, som hurtigt kan give et indtryk af trusselsbilledet i vores vejledning. Kilde: [10]

4.2.4 Risikoanalysen og score

For at følge risikoanalysens trin fra figur 5 skal der efter identifikation af værdier/aktiver (det er identifikationen af services) og identifikation af sårbarheder (det er identifikationen af de for-

bundne inputs, fordi de "åbner" op til disse services), laves en vurdering af risikoen for angreb og sikkerhedsforanstaltninger mod disse.
Dette forudsætter dog, at der på ingen måde er åbent for kompromittering efter inputsne (dvs. på "indersiden" af produktet); det kan vi nok ikke garantere fuldstændigt, men idéen er netop, at fordi vi også har alle kablede forbindelser (inkl. ting som servicestik og strømpins) med i vurderingen og ser bort fra deres interne forbindelser (som formentlig er noget anderledes end produktets primære tilslutninger), så er vi dækket ind. Er det ikke tilfældet, er det en svaghed for modelleringen. Havde vi det imidlertid ikke med, ville den resterende del af modelleringen og de generaliseringer vi kan opstille med denne, blive væsentligt mere komplekse.

Umiddelbart vil den bedste måde at gøre dette på være at dele trinnet op i tre deltrin, som følger opbygningen fra trussels- og risikomatrixerne:

Konsekvens Første deltrin skal gå ind og se på hvor kritisk hver funktion er, altså hvor stor konsekvensen ved en kompromittering vil være. Der skal kigges på henholdvis data i funktionen og brugen af funktionen — både hvad angår modificering og ændring af funktionen. Ud fra disse overvejelser skal der gives en *score* til hver funktion. Der kan umiddelbart ikke laves en fuldstændig vejledning til hvilke funktioner, der skal have hvilken score, da det afhænger meget af produktet (en kompromittering af hukommelsen i en kaffemaskine, hvor eneste data er hvornår der er brygget kaffe er ikke helt så problematisk som en harddisk i et Smart TV, hvor brugerens Facebook-cookies kan ligge).
For at stimulere analytikerens tanker om konsekvensen af en kompromittering af de enkelte inputs, vil vi stille nogle overvejelsesspørgsmål. De velkendte CIA-properties (tabel 1) kan vendes om til spørgsmål, som kan afdække, om de er opfyldt og samtidig har vi de tidligere beskrevne spørgsmål fra trusselsmodelleringen i [23].
Scoren for de enkelte funktioner skal vi sørge for at sprede ud, så alle de funktioner som har forbindelse til hinanden, også har den samme score! Det er nødvendigt, idet en kritisk database på en lagerenhed, stort set altid vil kunne både læse og skrives fra af de andre forbundne funktioner og derved kan angribere potentielt bruge disse andre funktioner som angrebsvektorer. Selvom det ikke er tilfældet og funktionen kun har læseadgang, formindskes truslen ikke væsentligt, fordi der stadig kan udtrækkes kritisk data og funktionen ydermere kan være sårbar over for ændringer i dens rettigheder.

Trussel I andet deltrin skal der ses på hvert enkelt input og analyseres på kendskabet til dette. Vi kan igen ikke gå ind og give en detaljeret vejledning til enhver form for input og der skal derfor laves en generel opdeling af de forskellige inputs. En måde at kategorisere dem på, ville være at kigge på hvert inputs historie og se hvad der er af kendte angreb og derudfra give en score. Det ville dog blive et stort arbejde (og en uoverskuelig vejledning) og idéen skal derfor ændres en smule.

I stedet vil vi kategorisere dem groft efter kendskab og angrebsmuligheder. Vi lægger alle trådløse inputs med en tilgængelig dokumentation (og muligvis tilgængelige angreb) i en kategori. Alle kablede forbindelser der kræver fysisk tilstedeværelse for, at en kompromittering er mulig, lægges i en anden kategori og "resten", altså mere ukendte, kablede inputs, lægges nederst. De trådløse, men "ukendte" inputs, vil vi dog beholde i mellem kategorien, for alene det, at man kan undgå at komme fysisk tæt på, gør, at man som angriber vil have større incitament til at forsøge sig her.
På denne måde behøver vejledningen heller ikke tage højde for de forskellige angrebstyper på hvert input og den er abstrakt nok til at tage højde for specialinputs, som f.eks. servicestik.

Sikkerhed Sidste del er så analysen af hvad der i forvejen findes af sikkerhed på hvert input. Producenten skal her give en vurdering af sikkerheden på de enkelte inputs og dermed produktets sikkerhedsperimeter. Ud fra dette tildeles igen en score, som afspejler sikkerhedsforanstaltningerne. Det er vigtigt, at vejledningen desuden indeholder noget om social engineering og evt. insider threats i et vist omfang (jf. 3.2), fordi det er blevet en meget nemmere angrebsvektor for angribere pga. de sikkerhedsforanstaltninger der er gjort de senere år mod digitale angreb [13]. Social Engineering forekommer i mange former, hvor listen på Wikipedia[27] [1], kan være et udmærket udgangspunkt til at guide brugerne i selve vejledningen. Det er i virkeligheden blot en udvidelse til tankegangen om, at hardware kun har behov for at blive beskyttet fysisk [18] og derfor præsenteres som en række forslag til hvad man kan overveje i den forbindelse.
Vores model skal dog ikke handle om hvordan man sikrer den fysiske organisation (som OCTAVE gør [20]), men at medtage disse mere håndgribelige trusler i det der formentlig bliver en oversigt over angrebstyper, kan lede til nogle nye overvejelser i det daglige, som nemt kan vise sig at være en fordel. Vi skal dog ikke lægge for stor vægt på det, da det er ud over vejledningens anvendelsesområde.

I ovenstående omtales en score til at differentiere resultaterne. I 3.3 omtaler vi DREAD, der er en måde at vurdere trusler på. Vi vil herudover også bruge trusselsmatrixerne til farvelægningen. I disse indgår også en score, hvor man som sagt tager produktet af den estimerede konsekvens og trussel og deler det med sikkerhedsforanstaltningerne.
DREAD vil kunne bruges på samme måde her, fordi vi kan farve ud fra scoren, men som beskrevet, så er metoden svær at bruge, fordi spørgsmålene dækker flere aspekter på en gang og er svære at vejlede i. For at forbedre undersøgelsen, vil vi derfor hellere bruge det traditionelle scoresystem, som følger af trusselsmatrixerne og benytte en skala, med relativt få trin, for netop at eliminere et af de andre kritikpunkter fra DREAD: At dens mange niveauer er for

[27] http://en.wikipedia.org/wiki/Social_engineering_(security)#Techniques_and_terms

komplicerede. I stedet agter vi at bruge en skala, hvor vi direkte kan differentiere mellem de forskellige trin og formidle dette i vejledningen. Navngivningen af de enkelte trin vil da være lig dem som foreslås i kommentarerne på [5]: Lav, middel og høj.

4.2.5 Forslag til sikkerhedsforanstaltninger

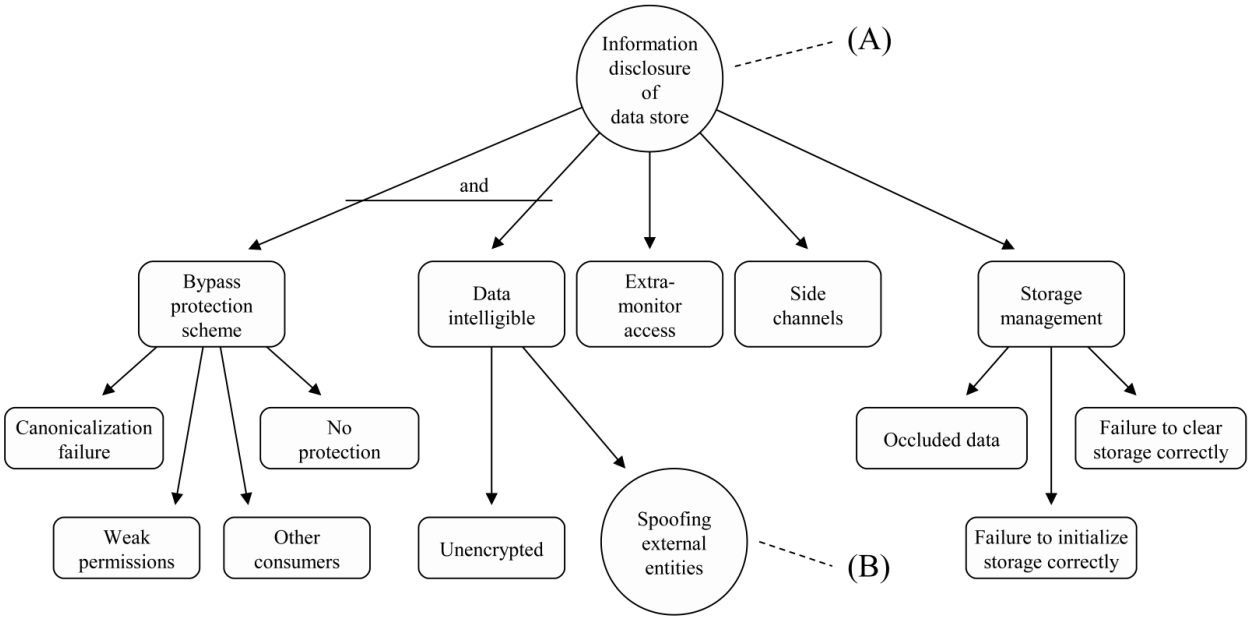
Endeligt skal metoden indeholde løsninger (eller i det mindste fingerpeg på sådanne), som kan sænke de identificerede risikoniveauer. Vi har in mente, at producenten på nuværende tidspunkt står med et komplet billede af hvor produktets formodede sårbarheder findes – og mens han givetvis har et vist kendskab til sikkerhedsprincipper (hvilket vi forudsætter for at kunne gennemføre analysen), så kan vi ikke antage, at der i virksomheden findes en ressourceperson, som kender løsninger på de sårbarheder, der er identificeret.

Ved trusselsmodellering med STRIDE, bruger man, som beskrevet i 3.3, *threat trees*. Det samme store threat tree bruges antageligt gennem Microsofts designfase af stort set alle produkter, så her ligger der en ufattelig stor mængde erfaring og ekspertise bag. Samtidig er den rettet mod software specifikt og ikke den hardware-/softwarekombination vi antager.
Det andet præsenterede værktøj er *attack patterns*, som er generelle beskrivelser af de trusler som findes mod grupperinger af forskellige komponenter i systemet.

De er særdeles brugbare til at afsløre de specifikke problemer, når de først er konstruerede, men vi kan imidlertid ikke konstruere attack patterns på den "rigtige" måde, for det vil være *alt* for omfattende at dokumentere alle former for angreb på alle systemer på denne måde[28].
I stedet vil vi prøve at opbygge en række forslag for hver del af systemet for hvilke typer af angreb, man bør forvente og hvad modtræk kan være mod disse. Det vil nok ikke blive fuldstændig dækkende, hvilket der skal tages højde for, men der kan f.eks. ydermere henvises til relevante værktøjer eller kilder, som producenten kan bruge til at scanne sit produkt. Dette skal være værktøjer sammenlignelige med de værktøjer som pen-testere bruger, hvorved producenten er hjulpet ganske godt på vej og vil være i stand til at finde samme information som størstedelen af sine mulige angribere, men samtidig, qua resultaterne af modelleringen, have et klart billede af hvor på produktet disse værktøjer skal anvendes.
Som nævnt i afsnit 3.3 vil vi dog aldrig på denne måde kunne bevise, at vi har dækket alt ind. Vi kan dog komme et godt stykke ad vejen med forslagene, fordi man ellers potentielt kunne komme ud for, at den ansvarlige for undersøgelsen helt mangler denne viden!
Vi kender også et generelt billede af sårbarheder som fra figur 2. Her illustreres generelle sårbarheder/angreb for software, hardware og data'en og det kan tjene som en let, overskuelig opdeling

[28] Et eksempel på hvor specifikke disse *attack patterns* laves, kan ses på https://buildsecurityin.us-cert.gov/articles/knowledge/attack-patterns/attack-pattern-generation

i vores vejledning. Udvidet med sikkerhedskriterne og deres ækvivalente trusler (tabel 1), kan vi dække de forskellige grupper af sårbarheder generelt.



Figur 7: Et eksempel på et threat tree pattern. Gengivet[29] efter [16]

I [16] foreslås en opbygning, som kombinerer disse to og afhjælper lidt af problemet med, at producenten ellers skulle konstruere et threat tree selv. For at gøre dette, skulle vedkommende, med en god portion kreativitet og viden i baghånden og med udgangspunkt i hver komponent, definere hvad der måtte opstå af trusler imod dette iht. STRIDE-tilgangen. Et eksempel herpå ses i figur 7, hvor der undersøges trusler imod læk af information ("Information disclosure" – STR**I**DE) af systemets data, hvilket i bladene på grafen resulterer i konkrete sårbarheder, som kan løses.
Problemet er imidlertid, at udover den store mængde viden og lange udviklingstid (som også passer dårligt til det konkurrenceprægede miljø, som vores målgruppe indgår i), så er der mange gentagelser mellem de forskellige STRIDE-trusler og komponenter og man bruger både et system-perspektiv (ser sårbarheder i systemet) og angrib-perspektiv (dennes handlinger) [16].

Den foreslåede løsning består af en komplet liste af trusler, et eksempel, modforanstaltninger og evt. afhængigheder (hvis andre dele af et threat tree også er nødvendige for udnyttelsen); se figur 8. Denne struktur passer bedre til vores formål.
Vi mener dog, at fordi vi målretter en hardware-/softwarekombination, er det nødvendigt at skelne lidt. Således vil vi tage udgangspunkt i opdelingen fra figur 2, når vi skal præsentere vores eksempler på modtræk. Afhængigheder og konkrete trusler kan også hurtigt bliver meget

[29]Oprindeligt fra "Howard, Michael and Lipner, Steve – *SDL: The Security Development Lifecycle*, Microsoft Press, 2006", men figuren kan ikke findes heri, så vi gengiver fra vores kilde

```
Threat
    Threat
        Example
        Mitigation
        Dependency
    Threat
        Threat
            Example
                Mitigation
            Example
        Threat
            Mitigation
            ...
```

Figur 8: En anden threat tree-struktur. Kilde: [16]

komplicerede og direkte modarbejde den generelle udformning, som vi skal opnå for at dække den brede målgruppe, så de udelades.
Vi bevarer dog den listeformede struktur fra figur 8, men indskyder, mellem vores komponent type og trussel, et sikkerhedskriterie. Således skal vores threat tree udformes omtrent som skitseret på figur 9.
Denne opdeling er, efter vores mening, både naturlig og nemt tilgængelig, hvilket [16] også understøtter. Vi fandt på dette design, *før* vi fandt kilden, hvilket igen understreger den simplicitet og naturlige opbygning denne form har.
Sikkerhedskriterierne baseres på STRIDE og figur 2, som supplerer hinanden og dermed også overlapper i nogle tilfælde. STRIDE er det bedste udgangspunkt, for at formidle en tilpas dækkende identifikation af trusler [16], så med inklusionen af figur 2 fra [21], bibeholder vi som minimum det samme niveau. Fordi vi differentierer imellem hardware, software og data, er det ikke sikkert, at alle sikkerhedskriterierne kan bruges på alle tre komponent typer.
De forskellige løsninger er eksempler, som vi primært henter fra vores almene viden. Desuden kan der suppleres fra [25], som bl.a. indeholder generelle råd om datasikkerhed, for at brugeren kan anvende dem i udviklingsfaserne.

Den konkrete implementering kan trække komponenter fra alle disse steder, men den lette, overskuelige opdeling fra figur 2 er meget tiltalende, fordi den også er nemt aflæselig. Derved kan producenten se på sin model, se på vores forslag til sikkerhedsforanstaltninger for de forskellige trusselstyper (taget fra figuren) og så selv deducere nogle modtræk, som samlet kan sænke truslen for hans system.

4.3 Design af vejledningen

Vejledningen vil blive opdelt i to hoveddele.
Den første del vil blive selve vejledningen med de fem trin, som en producent vil gennemgå på

```
Component type
    Security property 1
        Example
            Mitigation
            Mitigation
    Security property 2
        Example
            Mitigation
    Security property n
        ...
Component type
    ...
```

Figur 9: En skitse af vores threat tree-udformning baseret på [11, 16, 21]

det pågældende produkt. Anden del vil blive en eksemplificering af vejledningen på de to gennemgående produkter i projektet: Dørlåsen og Smart TV'et, hvor vi ligeledes vil kunne fremvise den modellering, som de foregående afsnit lægger op til.
Valget er faldet på disse produkter, delvist fordi de er så forskellige og delvist fordi vi kunne låne dørlåsen som demoudstyr. Forskelligheden er en stor fordel, fordi vejledningen netop er rettet mod en så bred produktgruppe, så uanset hvad, er det en nødvendig del af den samlede vejledning, at der findes nogle eksempler, som kan demonstrere metoden i praksis.

Vejledningen vil grundlæggende følge de samme trin, som vores to referencemodeller lægger op til. Disse beskrives i 3.3.
Succeskriterierne er beskrevet i afsnit 2, hvor det især er vigtigt for os, at vejledningen kan bruges af et team med en mindre datasikkerhedsmæssig baggrund end vores.

4.3.1 Generel form

Grundlæggende betyder det, at vores vejledning på samme måde skal være bygget op om de tre trin af stigende "kompleksitet", men hvor man i STRIDE får seks konkrete typer af sikkerhedsbrister, som man skal analysere for og hvor OCTAVE reelt ikke giver nogen konrekte fingerpeg, vil vi hellere prøve at gøre det helt generelt og på et højere niveau, for så at lade det ansvarlige team foretage den konkrete vurdering ud fra disse retningslinjer. Ift. STRIDE undgår vi således at give analytikeren skyklapper på, men peger samtidig vedkommende i den generelle retning, således at rangeringen af de forskellige værdier i produktet laves bedst muligt og med udgangspunkt i produktets virkelighed.
Målet er at opnå den rette balance mellem vejledning og selvstændig tænkning, hvor vi antager en "gylden middelvej".
I risikovurderingen vil vi, dog med enkelte fingerpeg, overlade arbejdet til analytikerne. Vi

antager en meget bred målgruppe, så i modsat fald skulle vi bevisligt kunne dække *alle* produkttyper på en generel måde og så bagefter også give fingerpeg om hvad der *kunne* udgøre risici for de enkelte elementer. Det vil blive for omfattende og med for stor risiko for mangler. Denne udførsel minder om de to andre modeller, som har samme tilgang.
Endeligt forsøger vi at lave en identifikation af angreb på en generel måde. Det er ulig de to andre metoder, hvor man igen forlader sig på det for evalueringen ansvarlige team. Forskellen er dog, at vi har flere ligheder på produkterne, som ydermere i det første trin (modelleringen af produktet) er blevet "homogeniseret" i form af inputs og derfor kan opstille modtræk bedre. Da vi har projiceret værdien af de enkelte services ud til deres inputs, har vi stiliseret det på en måde, så man kan nøjes med at vurdere sikkerheden her (undtagelse er dog, som diskuteret ovenfor, hvis inputsne ikke kan modelleres som "single-point-of-failure" på denne måde).

Princippet for modelleringen bliver at analysere produktet udefra og så arbejde os indefter. Det er irrelevant at kigge på services der ikke kan nås udefra, eller hvor de redskaber der i så fald skulle bruges, ville kræve en langt større indsats, end andre indgange på produktet. Vi ved også med sikkerhed, at der er bedre og flere tilgængelige angreb mod mere brugte inputs som WiFi end der er imod infrarød og lignende. Vi starter altså med at analysere produktets input og som beskrevet i afsnit 4.2.2 kategoriserer vi dem i trådløse og kablede input. Dernæst går vi et skridt længere ind i produktet og analyserer hvilke services der findes og hvilke input der er forbundet til hvilke services.

Det er generelt vigtigt i udarbejdelsen af vejledningen at gøre brugeren opmærksom på at være selvkritisk. Det hjælper ikke at feje sikkerhedsproblemerne ind under gulvtæppet ved gennemgangen, så dette vil vi opfordre yderligere til, ved i hvert trin at lave en kort metatekst, som motiverer og forklarer bevæggrunden for det givne indhold af trinnet. Det skal være en "light"-udgave af indholdet fra rapporten, hvor producenten også kan få en lidt dybere indsigt i problemet (som han evt. kan bruge til at tage problemerne op på ledelsesniveau). Dette hidrører også af det sidste succeskriterium for rapporten, om at den skal være "anvendelig". For at den virker sådan for brugeren, skal vedkommende kunne se fordelen ved at investere tiden i udarbejdelsen; sammen med introduktionen til vejledningen, sikres det på denne måde.

4.3.2 Prioritering og udbedrelse af sårbarheder

Derfra skal der udarbejdes en prioritering af hvilke inputs og services, som har et reelt sikkerhedsproblem og hvilke som ikke er kritiske. Dette kan gøres ud fra en vurdering af hvilke services, som er essentielle for produktet og hvilke der er *gimmicks*. Det skal samtidig også gøres klart, at selvom vejledningen følges til punkt og prikke, vil der altid være mulighed for nye og uopdagede sikkerhedshuller og den kan derfor ikke bruges som et endegyldigt bevis på,

at der ikke eksisterer sikkerhedshuller i det pågældende produkt, men snarere et værktøj til bekæmpelse af disse.

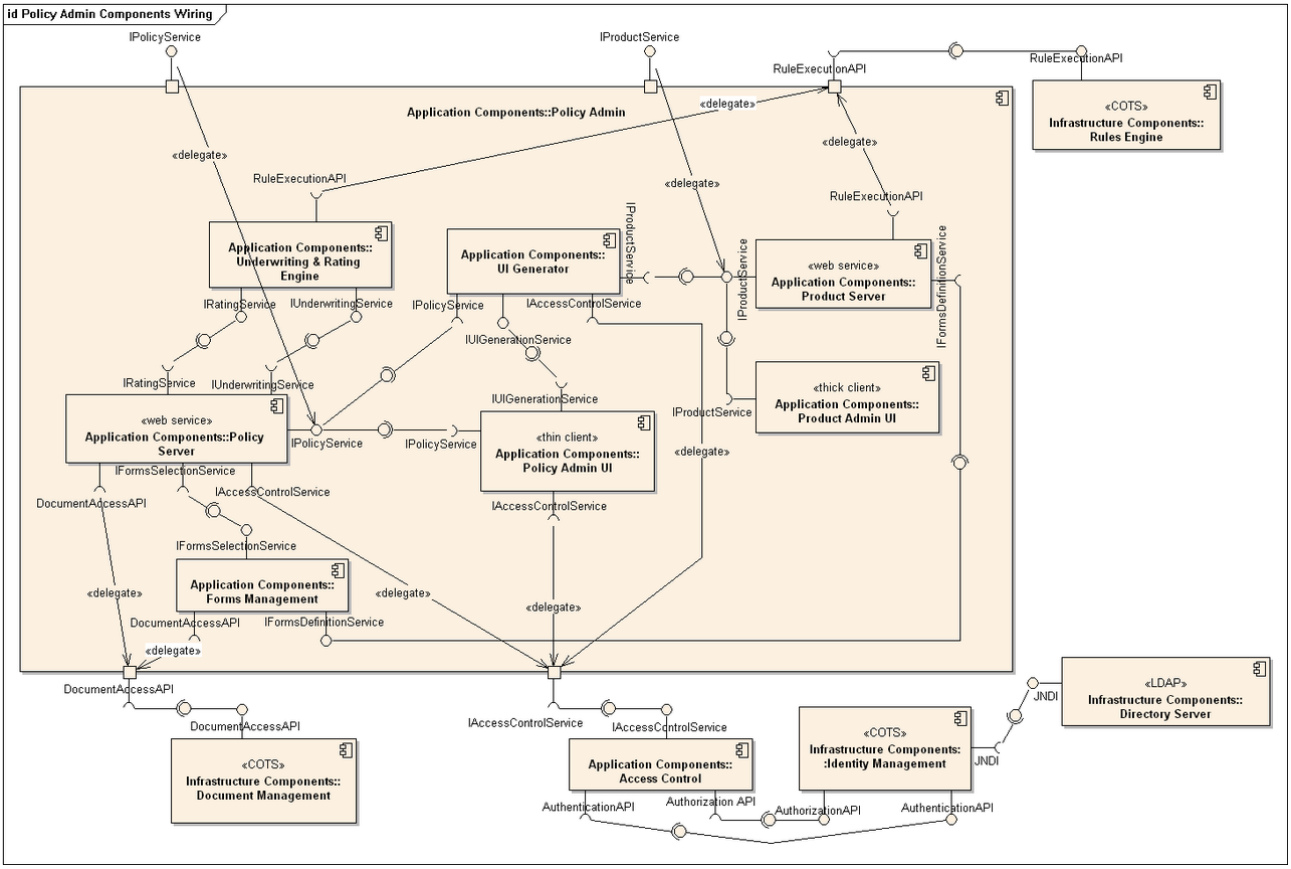
4.3.3 Modelleringen

Eksemplerne skal være tydelige og det skal gøres tydeligt, hvordan de lægger sig tæt op ad vejledningen. Som beskrevet i afsnit 4.1, er figurer og farver med til at hjælpe forståelsen hos mange. Det er derfor også vigtigt, at vores figurer er lette at afkode og forstå, hvilket vil medføre, at vejledningen virker lettilgængelig og brugbar.

I det hele taget er det vigtigt, at vi i implementeringen af vejledningen, tænker over hvordan vi bedst faciliterer producentens undersøgelser! Vi tror, at overvejelserne om den klare kommunikation (med farver, figurer og præcis tekst) er en af de vigtigste faktorer hertil, for når man stiller en person over for mange eller komplicerede valg [19], så forvirrer det os, hvilket igen gør analytikeren mindre motiveret til at give sig i kast med opgaven; vejledningen udvikles jo for at blive brugt, så usability er bestemt vigtigt.
Overvejelserne omkring dette vil også være relevant at opridse i introduktionen til vejledningen. Denne skal ydermere også indeholde en klar fremgangsmåde, hvor der bl.a. understreges, at processen er dynamisk og at det er vigtigt, at man kan gå tilbage, hvis man opnår ny viden undervejs. Dette er også en gængs tilgang indenfor risikomangement (jf. 4.2).

Samtidig skal vi benytte kendte elementer fra faget. Udover de allerede nævnte, vil vi lave modelleringen vha. et UML[30]-lignende diagram. Vi skal selvsagt ikke have use case- eller klassediagrammer her, men opbygningen af et component diagram (som i figur 10) minder meget om den løsning vi forestiller os.
Her er de interne processer samlet indenfor perimeteren af komponenten og det er tydeligt hvilke dele af koden, der eksponeres udadtil og giver adgang til denne [2].

[30]Et "visuelt" sprog, som bl.a. bruges til at beskrive opbygningen af software. Se: http://en.wikipedia.org/wiki/Unified_Modeling_Language



Figur 10: Eksempel på et component diagram. Kilde: [2]

5 Vores metode

Forfatter: Rasmus

I dette afsnit gennemgår vi selve udarbejdelsen af vores vejledning og de valg der er taget ift. analysen (afsnit 4). Desuden kigger vi på "resultaterne" af projektet, hvilket konkret er de to eksempler på modelleringen, som også indgår som eksempler i selve vejledningen og hvordan disse går i tråd med vores forventede, færdige produkt.

5.1 Udarbejdelse

Vores vejledning bliver delt op på en måde, som følger de i teorien gennemgåede elementer. Det betyder, at der først opbygges en model af systemet, dernæst en risikoprofil og endeligt foretages foranstaltninger mod disse.
Den bliver opdelt, så vejledningen kommer først og selve modelleringen bagefter. Det er valgt fordi vejledningen som sådan skal kunne stå for sig selv og fordi modelleringen og figurerne fylder en del, så vejledningens (forsøgte) korte og præcise sprog, modarbejdes. Der gives så henvisninger fra hvert trin, som også korresponderer direkte med en ny tilføjelse på modellen og derved er den direkte sammenhæng bevaret.

5.1.1 Vejledning

Vejledningen bliver indledt med en formålsbeskrivelse om hvordan den skal bruges. Som vist i 3.3, vil man her typisk udarbejde succes- og sikkerhedskriterier, men vi mener, at det vil kræve en dybere gennemgang af sikkerhed og hvad det indebærer, hvilket *ikke* hænger sammen med den datasikkerhedstekniske forudsætning, som vi har antaget for analytikeren.
I stedet bliver udarbejdelsen lagt ned i det relevante trin af selve vejledningen, som diskuteret i 4.2. Det gør det også mere sandsynligt, at vi får en modellering af produktet som er mindre farvet af producentens holdninger, så der ikke lægges antagelser om sikkerheden på de enkelte inputs eller services ind over.
En anden mulighed er, at have dem i starten og gøre opmærksom på problemer inden systemets indhold/funktioner gennemgås. Dermed opnås indsigten fra starten af, så producentens syn kan ændres og vedkommende kan hjælpes til at se bredere på systemet, når det skal modelleres. Vi tror dog ikke trods alt ikke, at producenten vil have så svært ved at tænke videre ift. sin eksisterende viden og samtidig er vores vejledning med den nuværende udformning, netop gearet til at forsøge at ødelægge eventuelle antagelser om sikkerheden, som man måtte have på forhånd. Det gør vi ved, at de første tre trin helt nøgternt bare oplister udformningen af systemet.

5.1.1.1 Trin 1 I det første trin modelleres systemet som en lukket boks med en "perimeter", hvor de forskellige inputs udgør forbindelsen mellem det indre og ydre. Derfor skal producenten først opliste disse.
For at gøre det nemmere, giver vi eksempler på hvilke typer af forbindelser, der kunne være og håber dermed, at der gives et heldækkende billede, hvilket, både her og gennem resten af vejledningen, er essentielt for, at produktet reelt kan bruges til noget i den sidste ende. Samtidig prøver vi med en kortere, motiverende tekst at fortælle, hvorfor det er vigtigt; man kender det måske selv: Hvis man ikke kan se meningen med noget, kan det være svært at lægge den rette energi i projektet – uanset, at man er ansat til dette; det vil vi gerne hjælpe med her og også prøve at lave en rød tråd i vejledningen, som giver mening uden en dybere forklaring. Samtidig kan noget af motivationen bruges til forklaring overfor en ledelse, som måske finder en sikkerhedsevaluering unødvendig.
Her er medtaget enkelte outputs, men som der også forklares i teksten, så er det fordi, at vi lige præcis på disse har fundet konkrete problemer, som *kan* ændre ved systemets funktion. Ved HDMI kan CEC-funktionen kompromitteres[31], mens f.eks. et DVI- eller VGA-stik også sender data om monitoren tilbage til systemet og således potentielt er sårbart[32] (som dog nok begrænses primært til tilgængeligheden).

5.1.1.2 Trin 2 Dernæst prøver vi at få listet alle de funktioner, som findes "inden i" produktet og kæde dem sammen med de forskellige inputs. Denne del er diskuteret meget, fordi det kan være svært at stillisere systemet på denne måde, når forbindelserne internt er væsentligt mere avancerede end blot at trække en ledning mellem de nogle bokse. Vi finder imidlertid samme fremgang i de andre modeller, hvor man vha. DFD'er også laver en sådan forenkling. Fordi vores formål er at lave trusselsanalyse og målet dermed er at vise, hvor der er størst risiko for sårbarheder, må vi se bort fra en dybere teknologisk kobling og i stedet fokusere på at vise, hvor truslen er størst. Ved at gå frem funktionsvis, gives muligheden for at tage fat i hver enkelt funktion og dens data og vurdere dette individuelt.
Vi bevæger os også her på et meget generelt plan, fordi systemerne kan være så forskellige; listen af eksempler kan synes kort, men den giver fingerpeg, som er tilstrækkelige for at få alt dækket [21].

5.1.1.3 Trin 3 Vha. state-/controlmodellen trækkes der dernæst linjer mellem inputs'ne. Modellen har været svær at definere på en god måde. Den beskriver tydeligt nogle forskelle mellem målgruppens systemer, men samtidig skal den bruges korrekt, for ikke at gøre mere

[31] https://media.blackhat.com/bh-eu-12/Davis/bh-eu-12-Davis-HDMI-WP.pdf

[32] Et spørgsmål på StackExchange.com her: http://security.stackexchange.com/questions/19007/vga-hdmi-based-attack. Vi har desuden læst andetsteds om brug af en VGA-pin, som returnerer information om monitoren, til et mindre angreb

skade en gavn. Vi præsenterer den derfor først i selve modelleringen.
Den er brugt til at anskueliggøre samspillet mellem controllerne og services og dermed give et overblik over hvordan produktets interne funktioner virker. Dette er et nødvendigt step for at få modelleringen af produktet til at være så præcis som mulig.

5.1.1.4 Trin 3b For at sætte særligt fokus på inputs som er tilrettede forbrugere og dem, som er helt skjulte, så vi bl.a. i 4.2.2 på at angive hvem, der er tiltænkt adgang til de forskellige inputs, for derved at et give overblik over dette.

Det blev dog besluttet ikke at bruge denne form for analyse – primært fordi brugbarheden er meget begrænset!
Analysen skulle have identificeret hvilke input, som var tiltænkt producent/tekniker/bruger og derved f.eks. kunne angive hvor "udsatte" de er. Imidlertid har vi ikke til hensigt hverken at nedjustere eller opjustere sikkerhedsrisikoen på baggrund af denne analyse, fordi man på den måde lægger nogle antagelser ind over undersøgelsen; at et input kun er brugt af producenten, vil i nogle tilfælde sikkert betyde, at det er mere "skjult" (og det skal vi se bort fra) eller at det har færre eller dårligere sikkerhedsforanstaltninger (og det skal undersøges særskilt). Selvom et input udelukkende er rettet mod forbrugeren, behøver vi ikke at tillægge det større opmærksomhed, men udelukkende behandle det på den neutrale, objektive måde, som gøres efterfølgende.

5.1.1.5 Trin 4 Med en model af systemet, går vi over til at opbygge risikoprofilen af produktet. Dette består af tre dele, der, som beskrevet i analysen, kommer fra datasikkerhedsfaget: Konsekvens, sandsynlighed og sikkerhedsforanstaltninger.
Analytikeren skal, helt subjektivt, analysere hvordan systemet er udsat for diverse trusler. Vi stiller spørgsmål til overvejelse, som er baseret på CIA-egenskaberne, OWASP og [8] jf. afsnit 4.2. Vi kan ikke på en tilstrækkelig generel måde tage stilling til, hvad der er de kritiske dele af systemet og den risikovillighed organisationen har og derfor er dette den bedste måde at få beskrevet det på. Samtidig er producenten også den der antageligt kender produktet og organisationen bedst og kan vurdere disse ting.
Vi giver ud fra dette en relativ score, der, som diskuteret i indholdsanalysen (4.2), skal holdes med et mindre antal differentierbare skalaer. Imidlertid har vi i første trin valgt en skala fra 1 til 5, fordi vi mener, at der kan være stor forskel på de enkelte funktioner imellem og, at man samtidigt som producent har et tilstrækkelig højt indblik, til at kunne udføre dette, så det vil vi gerne afspejle.

I vurderingen af sandsynligheden og sikkerheden holder vi os dog til en tretrinsskala, fordi vi her bevæger os over i en mere datasikkerhedsteknisk del af analysen, hvor vi ikke kan antage

samme viden. Derfor giver vi også en helt specifik skelnen mellem de forskellige niveauer jf. kommentarerne på [5] og afsnit 4.2.4.
I modelleringen af ovenstående udregner vi en score på en DREAD-lignende måde. Selve funktionerne og deres inputs farves for hvert trin og dermed kan resultatet (effektivt; se afsnit 4.1) aflæses direkte fra modellen. Således bliver den et meget stærkt værktøj, hvor man, både i ledelsen og i udviklingsafdelingen, kan se hvor man bør sætte ind.

Vi har overvejet hvordan vi kunne få sikkerheden til at veje tungere, idet vi gerne vil kunne afspejle eventuelle ændringer man måtte lave som modtræk, men samtidig må vi holde fast i, at modelleringen viser hvor det er sandsynligt, at der findes trusler mod systemet og ikke reelle sårbarheder. Vi skal samtidig lave det generelt, så det kan passe til både højteknologiske systemer med WiFi og bluetooth og mere lavpraktiske og "dumme" produkter med mindre avancerede protokoller. Løsningen blev dermed også at holde fast i vores differentierede skala på 1 til 3.
Den præcise graduering af truslen mod inputs efter hvor *dokumenterede* de er og den fysiske nærhed et angreb kræver, kan være vanskelig at vurdere. Vi arbejdede tidligere med en skala, hvor alle fysiske inputs fik scoren 1 og det kun var de trådløse inputs, som skulle gradueres efter hvor "kendte" de er, men vi indså, at der var stor forskel på en USB-forbindelse og antenneindgangen ift. den sandsynlige trussel mod dem.

5.1.1.6 Trin 5 Endeligt udvikles en vejledning for analytikeren, som kan bruges som *udgangspunkt* for at forhøje sikkerheden i systemet. Vi undersøgte både *threat trees* og *attack patterns* som et værktøj hertil, idet de kan bruges beskrivende til både at identificere og siden koble løsningsforslag på de enkelte sårbarheder [16]. Imidlertid er arbejdet med at opbygge disse for stort og out-of-scope, så løsningen, hvor vi ud fra hhv. hardware, software og data beskriver en række trusler og mulige løsninger, giver en bedre, generel tilgang, som i øvrigt stemmer overens med vejledningens generelle natur.
Vi afviger dog fra den i 4.2.5 foreslåede struktur, idet vi har brugt truslerne, som korresponderer med de forskellige sikkerhedskriterier (se tabel 1), til at beskrive de forskellige typer af sårbarheder mod delene. Vi mener, at det gør truslerne mere håndgribelige og forståelige for producenten, fordi vi fokuserer på et problem og hvordan det kan løses i modsætning til at have et, mere eller mindre, arbitrært mål, som komponenttypen skal opfylde.

Vi præsenterer, for hver trusselstype, en række konkrete løsninger, som *kan* afhjælpe det givne problem eller minimere risikoen. Vi ved godt, at listen bliver generel og ufuldendt, hvilket der således også gøres opmærksom på i vejledningen.
Forslagene er taget dels fra vores eksisterende viden fra både job og studie og dels fra relevante

kilder (især [8, 23], men også anden litteratur læst de seneste år). Vi har haft fokus på at samle en generel mængde af forslag, som kan anvendes på flere typer produkter. Imidlertid er det i sagens natur svært at ramme alle de mange forbrugerprodukter, for hvem vi tænker, at vejledningen har relevans, men vi har forsøgt at tænke over flere muligheder ud fra kig på et bredt udvalg af produkter og håber således at have dækket fornuftigt, men vi har intet kvalitativt mål herfor.

5.1.2 Modellering

Vha. en UML-lignende model med en perimeter, hvor pile markerer de enkelte inputs, som skaber forbindelsen mellem forbrugeren og produktets indre.
Selve opsætningen af modellen og placeringen af de enkelte funktioner som bokse inden i, er som sådan ganske ligefrem. Imidlertid har vi haft problemer med at inkorporere state-/controlmodellen på en brugbar måde i modellen.
Den har sin berettigelse som hjælpemiddel ift. bestemmelse af funktionernes forbindelse, men med vores fokus på de enkelte funktioners værdi og deres forbindelse med de forskellige inputs, bliver de uafhængige komponenter som RAM/ROM irrelevante. De er medtaget i de eksempler vi har lavet for vejledningen, men ikke yderligere forklaret, så man kunne, i en anden iteration af udarbejdelsen af vejledningen, ændre dette og finde en anden måde at modellere lige denne del på.
Vi har diskuteret andre muligheder; f.eks. at helt se bort fra modellen, men i et produkt, hvor man har en central styring og hvori al kommunikation mellem inputs og services går igennem denne, vil vi således ikke have formået at projicere de potentielle sårbarheder korrekt ud på inputsne.

Vi har desuden indført afgrænsningslinjer for at skelne mellem de tre dele af modelleringen: Controllers og deres inputs, en eventuel koordinerende styring og de forskellige services. Det skaber et mere ordnet billede, som igen bliver nemmere at afkode.
De forskellige controllers er en måde at sørge for, at inputs, som både har en kablet og trådløs forbindelse, kommer til at dele score. Det skal ses i lyset af, at vi ikke med garanti kan sige, at sikkerheden i systemet udelukkende afhænger af selve inputtet og dets implementerede sikkerhed, for det kan også være påvirket af f.eks. en produktspecifik håndtering af datapakker! Derved vil det være forkert ikke at kalde ethernet-forbindelsen ligeså usikker som WiFi.
Omvendt må vi erkende, at vi ikke har fokus på sikkerheden inde i selve controlleren og hvad der ellers sker imellem input og funktioner (evt. central styring), så ligeledes kunne en anden iteration af vejledningen, yderligere udvides til at se bort fra disse for simplicitetens skyld. En nærmere undersøgelse må dog til, for det kan være, at det kommer til at ske på bekostning af korrektheden af hele modellen, hvilket ikke kan accepteres.

Opdelingen med controllers, styring og funktioner kommer af en tidligere iteration, hvor vi havde en opdeling i controllere, primære funktioner og sekundære funktioner. Denne opdeling var imidlertid dårlig, fordi vi kan skabe nogle antagelser om, at funktioner, som ligger blandt de sekundære funktioner, er mindre udsatte eller vigtige for systemet som helhed – en ting, som vi skal undgå og som metodikkens opbygning ellers netop skal hjælpe med at fjerne.

Iht. 4.1 har vi farvet de forskellige funktioner og inputs med en farveskala, som signalerer, at jo højere score, desto "farligere". Den rangerer fra en mørkegrøn for laveste score til en mørkerød for højeste, således at man hurtigt kan se, hvor de mest kritiske inputs findes. Udregningen for gradueringen for skalaen 1-15 kan ses på figur 24 i appendixet (sektion B). Vi har opdelt det, så de fem farver får lige stor andel, men herefter har vi flyttet det ét step opad. Derved sætter vi større fokus på de særligt kritiske dele af modelleringen. Samtidig vil produktet af scoren sjældent blive helt i toppen, så derfor giver det også mening at rykke det et enkelt trin. For skalaen på trin 4.1 (fra 1 til 5), korresponderer de fem farver direkte 1:1.
Udover dette, består modellen som sådan blot af linjerne mellem funktioner/services og input, hvilket er med til at gøre den relativ enkel og ligetil at udforme for både os og producenten.

6 Resultater

6.1 Brug af metoden

I de to følgende afsnit kigger vi på hvordan vores vejledning anvendes i praksis på AccessZone dørlås hhv. et Samsung Smart TV. Vi har i forvejen brugt de tegninger som er produceret hertil, for at illustrere anvendelsen i praksis overfor brugeren af vejledningen. Herunder forklarer vi hvad der er tænkt herom og hvad der er gjort af antagelser.

6.1.1 AccessZone

Forfatter: Daniel

For at få evalueret brugen af metoden, har vi brugt denne til at modellere og analysere et AccessZone-dørlås stand alone system. Selve modellen kan ses i A.3, men konklusionen er, at det mest kritiske ved AccessZone-systemet er bluetooth-input'et som muligvis kan kompromitteres.

Trin 1 Ud fra vores viden om produktet, både ved læsning af de tekniske specifikationer, præsentationsmaterialet [17], samt erfaringer ved demoforsøg, er det ikke vanskeligt at identificere de inputs som findes på dørlåsen. Vores umiddelbare vurdering er, at strøm- og servicestik er sammenfaldende, men for overblikkets skyld indsættes de hver for sig, som det ses på figur 15a. Derudover findes kun bluetooth.

Trin 2 De indvendige funktioner tilføjes til modellen — og da vi har vurderet at styringen af dørlåsen er central, har vi tegnet en central styring på modellen (jf. figur 17a). Mængden af services er begrænset, da dørlåsen ikke har en bred vifte af funktioner, men "kun" skal sørge for at holde uvedkommende væk fra området, enten vha. den fysiske lås eller ved at sætte en alarm i gang.

Trin 3 Da styringen er dørlåsen er central, vil alle controller og services være forbundet til denne (figur 18a). De tre interne services som vi har identificeret i dørlåsen må være forbundet, da både låsedelen og alarmdelen skal bruge informationer om brugerne og fordi alarmdelen skal bruge information om hvorvidt døren er låst eller ej.

Trin 4 Dernæst tilføjes en score til hver service, mht. hvor kritisk den er. Det ses tydeligt at dørlåsens services er mere kritiske end TV'ets. Det skyldes primært, at hvor TV'ets hovedopgave er underholdning, så skal dørlåsen holde uvedkommende væk. Dette og samspillet mellem komponenterne i dørlåsen (den centrale styring) gør, at en enkelt fejl vil være kritisk for hele systemet. Derfor er den røde farve altså trukket med helt op i modellen af dørlåsen.

Herefter laves en vurdering af hvor kritiske de tre inputs er. Vi har vurderet, at kendskabet til servicestikket er minimalt. Bluetooths standardkryptering er derimod forholdsvis svag og derudover kræver det ikke det store at finde værktøjer til at opsnappe bluetoothforbindelser, jf. afsnit 6.2.1.

Til slut skal sikkerheden på hvert input vurderes. Dette trin er for os en del antagelser, da vi ikke har dybdegående kendskab til produkterne.
På AccessZone har vi antaget, at der udover bluetooth standard kryptering også er lagt et ekstra lag kryptering, som gør at den kan tildeles en sikkerhedsscore på 2. Dette er gjort fordi det er tilrådeligt at lægge ekstra kryptering på bluetooth, især når det bruges til så følsomme ting, fordi sikkerheden i protokollen er så dårlig. Samtidig har vi også i vores forsøg på at bryde AccessZone-låsen kunnet konkludere, at der netop var dette ekstra lag kryptering.
Når vejledningen er fulgt og sikkerhedsscoren udregnet, er det dog stadig bluetooth-forbindelsen som står som den mest kritiske del af AccessZone-systemet. Dette var at forvente, både på baggrund af den meget begrænsede mængde input der findes i systemet, men samtidig også forbi bluetooth er sårbart og kendskabet til det er meget stort.

Trin 5 Der skal nu laves en grundig analyse af hvilke dele af produktet der skal prioriteres i forhold til sikkerhed. Eftersom bluetooth-forbindelsen har den højeste risikoscore, må vi antage, at det er her, der skal prioriteres. Ud fra vejledningens afsnit A.2.5 er det vigtigt at gå systematisk ind og se på de forskellige dele, som kan nås med bluetooth. Da produktet er med central kontrol, må vi antage at alle funktioner og services kan nåes med bluetooth og det er derfor kritisk, at der bliver tænkt sikkerhed i hele produktet. En god krypteringsalgoritme, med en fornuftig nøgleudveksling er en god start. Et forslag til dette kunne være at bruge en asymmetrisk krypteringsmetode til nøgleudveksling og dernæst bruge en symmetrisk til resten af kommunikationen. Denne metode er meget udbredt [8] og bestemt anbefalelsesværdig.

6.1.2 Smart TV

Forfatter: Rasmus

I det følgende gennemgås brugen af vejledningen og modelleringen af et Samsung Smart TV vha. metoden (afsnit A.2) og modelleringen (afsnit A.3).

Trin 1 For at identificere alle inputs i TV'et, har vi først og fremmest kigget på en produktspecifikation for Samsung Smart TV-familien, som er fundet online[33].
Desuden har vi kontrolleret for yderligere inputs i den liste som er kompileret i afsnit A.2.1. Det resulterer i de inputs, som ses på figur 15b.

[33] http://downloadcenter.samsung.com/content/UM/201303/20130316093517079/[DAN]X12DVBEUF-0313.pdf

Vi har her grupperet vores videoinputs for simplicitetens skyld – først og fremmest fordi der er så mange, men også fordi, at det varierer hvilke inputs, der præcis findes på de enkelte modeller og det langt fra er alle, som der findes angreb imod (et output som S-video kunne vi eksempelvis ikke finde angreb via). Da vi ser bort fra et simpelt video output som direkte datalæk, udelades disse.

Trin 2 Dernæst tilføjes de indvendige funktioner som på figur 17b. Vi er her begrænsede til nogle temmelig generelle services qua vores viden om systemet; som producent vil man dog kunne specificere disse bedre og eventuelt mere i detaljer.
Vi ser, at der gøres brug af state-/controlmodellen, hvor vi (antageligvis) ikke har en "koordinerende" funktion på samme måde som f.eks. i dørlåsen. Dermed fremstår de forskellige komponenter som selvstændige enheder.
Vi har inkluderet internetbrowseren som en selvstændig funktion, fordi den tænkes at være udviklet på stærkere principper end andre apps, fordi den eksempelvis kan inkorporere dele af nogle af markedets open source-browsere. Imidlertid er man ikke stærkere end sit svageste led og derfor er forbindelsen til app-platformen stadig markeret, idet den vil blive ramt af eventuelle sårbarheder i den bagvedliggende platform.

Vi finder, at der ift. Smart TV'ets kompleksitet ikke er så mange services som forventet, men har ikke (vha. produktspecifikationen fra forrige trin) kunnet identificere flere, når det samtidig skal holdes på et højere, abstrakt niveau. Var man gået længere ned og havde eksempelvis medtaget forskellige databanker og deslignende, ville vi først og fremmest øge vores risiko for at ramme forkert, men omvendt kunne man have lavet en mere fyldestgørende modellering. For et så specialiseret produkt som dette, vil vi med en udenforståendes kendskab ikke kunne gå nærmere.

Trin 3 Nu tilføjes forbindelser mellem de forskellige controllers, inputs og services (figur 18b). Fremgangsmåden er meget ligefrem og skal afspejle de forbindelser som er mellem de forskellige services, inputs og deres controllers. Vi ser, som beskrevet, på hvilke services der kan styres fra det enkelte input og kan derved få et relativt klart billede af hvordan de forskellige funktioner kommunikerer.
Desuden forbindes de services, som kommunikerer internt, således, at scoren i næste trin projiceres fra den højest scorende over på den anden for at afspejle denne forbindelse.

Som producent ville man uden tvivl (igen) have bedre forudsætninger for at lave denne analyse, da en "skjult" forbindelse (software deaktiveret) mellem f.eks. WiFi og en kritisk service, vil være en mangel ikke at få afdækket. Imidlertid må vi håbe på, at man som producent er klar over den risiko der introduceres herved og at man minimerer antallet af sådanne konstruktioner.

Vi har i øvrigt ikke set tegn på dette i de to produkter, vi har modelleret!

Trin 4 Endeligt indføres selve sikkerhedsevalueringen i modellen.
Først laves en vurdering af hvor kritiske de enkelte services er (figur 20b). Vi har her fire services, som vi graduerer efter vores subjektive vurdering iht. de spørgsmål vi stiller i A.2.4.1. Internetbrowseren vurderes til middel jf. forklaringen i trin 2 herover. Tuneren får samme graduering, fordi den, selvom den ikke indeholder (forbruger-)kritisk data som sådan, så udgør en væsentlig del af TV'ets funktioner, hvorfor det vil være uacceptabelt, hvis den ikke er tilgængelig.
Gradueringen projiceres nu mellem funktionerne internt og således bliver både app-platformen og browseren også 5 (røde), fordi de har forbindelse til den højkritiske harddisk – en forbindelse som potentielt kan udnyttes begge veje.
Opad får controllerne herefter de nødvendige farver fra funktionerne.

Dernæst laves vurderingen af de enkelte inputs, hvor vi iht. A.2.4.2 opdeler inputs efter den sandsynlige trussel imod dem. Det giver billedet som på figur 21b. Der er igen en del antagelser inde over, hvor man eksempelvis kan diskutere om 3D-brillen og voice control reelt er trådløse forbindelser og også om de kvalificerer sig til at få deres score ganget med 2.
Billedet er dog ikke komplet før end de nuværende foranstaltninger tages med. Det er sket på figur 22b.
Nu er billedet et lidt andet, for bedre sikrede protokoller, gør, at nogle af de kendte inputs kan nedprioriteres til fordel for f.eks. bluetooth, som i alle implementationer er usikkert. Her ser vi også årsagen til, at vi har ændret farveskalaen lidt med hensyn til de lavere værdier, for de røde inputs er helt forsvundet nu. Det er dog klart for producenten pba. introduktionen til vejledningen, at dette primært skal bruges som en rettesnor hvad angår prioriteringer og ikke som et resultat, som viser, at TV'et generelt er fuldkommen sikkert.

Trin 5 Der præsenteres nu en række foranstaltninger, som vi i teorien efterfølgende ville skulle implementere for at hæve sikkerheden i TV'et.
Med vores faglige udgangspunkt ville vi lave en prioritering af hvilke inputs, som kræver den største opmærksomhed og så gennemgå de forslag, som er listet i afsnit A.2.5.1. Især bluetooth og de nyere, men datasikkerhedsmæssigt ukendte teknologier som voice control og 3D-brillen, ville få et eftersyn iht. de oplistede principper, mens vi i første omgang ville springe over den infrarøde forbindelse, fordi den er relativ begrænset i sine muligheder for interaktion. Det er dog i høj grad værd at kontrollere, at den ikke pga. f.eks. en forkert implemementation, kan ændre ved nogle af de services, som vi tidligere identificerede den til at have forbindelse med.

En teknologi som WiFi og NFC scorer i den høje ende af den lysegrønne skala og vil som

sådan også være at betragte som usikre inputs, idet de er forbundne til services med en høj konsekvens, men modelleringen viser imidlertid noget andet.
Dette giver anledning til efterrationalisering hos os, for når vi kigger på produktet som helhed, ville vi graduere disse to inputs højere. NFC er også vurderet til at være helt uden sikkerhed, men pga. den vidstrakte skala for den endelige vurdering, bliver den alligevel kun sat på næstlaveste trin! Dette er utilsigtet, så det ville give mening at finde en løsning på dette i næste iteration.
Nogle af de foreslåede modtræk i A.2.5.1 vil også kunne anvendes direkte på funktionerne og skabe yderligere barrierer her. Det er noteret i vejledningen, at man kan det, men det er ikke en ting, som ændrer ved selve modelleringen.
Konkrete foranstaltninger kan vi dog ikke komme op med, fordi denne analyse bevæger sig på et relativ teoretisk plan.

6.2 Metodens resultaters videre anvendelse

Forfatter: Rasmus

Formålet med metoden er at skabe et godt overblik over sikkerheden i et system. Det gode overblik er nødvendigvis det overblik, der som minimum tilsvarer det en angriber kan skabe sig vha. sine værktøjer og sunde fornuft.
I det følgende vil vi derfor undersøge, hvordan man som pen-tester i praksis kunne gribe analysen an for et system i vores målgruppe og hvordan det harmonerer med metodens resultater.

Metoden giver producenten et svar på, hvor *sandsynligheden* for sårbarheder er størst. Det er samme fremgangsmåde som benyttes i de modeller vi har sammenlignet med (se afsnit 3.3, for i praksis vil det ikke være muligt at finde helt konkrete sårbarheder uden direkte at scanne efter hver enkelt.
Det er problematisk, at man for en garanteret identifikation af en sårbarhed, er nødt til at finde et proof-of-concept og derfor er det også et stadig forskningsområde, hvor de omtalte *threat trees* udgør et væsentligt værktøj [16].
Ser vi tilbage på afsnittet i teorien om angreb (3.2), så er processen på figur 1 ret typisk herfor [23]. For en ondsindet hacker, ville processen naturligvis begrænse sig til udnyttelsen af sårbarhederne, hvorefter hans mål vil være opfyldt. De enkelte trin kunne i praksis forløbe nogenlunde som her:

Planlægning De fleste "betalte hackere" (pen-testere) holder først et møde med kunden, hvor de aftaler målene og rammerne for testen (bl.a. i hvilket omfang testeren må tilgå eller modificere data). Er testeren ikke bestilt, går man i stedet direkte videre til næste trin.

Identifikation af sårbarheder Her indsamles information om det system som skal testes. Det kan f.eks. ske via læsning af produktblade, erfaringsudveksling med andre i branchen eller social engineering. Hos firmaets ansatte findes givetvis den største mængde information, hvorfor det er utroligt vigtigt, at man som virksomhed er klar over de trusler man kan blive udsat for. Dernæst kan man udføre detektering af potentielle sårbarheder vha. automatiserede værktøjer som Nmap[34] (til afdækning af services og deres eventuelle sårbarheder på en eller flere hosts) eller spot checks mod udvalgte dele.
Specialdele af systemer, som dem vi bl.a. dækker i de systemer vi har i målgruppen, vil en pen-tester typisk kun i ringe grad kunne anvende ovenstående værktøjer imod, fordi de eksempelvis enten slet ikke implementerer de kendte protokoller, kun har det som sekundære services eller bruger dem på en atypisk måde!
Her vil testerens erfaring og indsamlede viden komme i spil, fordi der skal tænkes alternativt ift. angreb mod computersystemer. Man kunne forestille sig, at sårbarhedstyperne vil være de samme set over en bred kam, men den specifikke manifestation af disse, vil afvige. Med de korrekte spørgsmål og social engineering-metoder, vil man imidlertid nok finde noget alligevel og derfor er det vigtigt, at producenten holder sine kort tæt ind til kroppen, for ikke at hjælpe angribere. Det skal dog samtidig være en afvejning ift. den hjælp man kan få fra et stort community, som kan hjælpe med at gennemgå ens kode.

Udnyttelse af sårbarheder Hernæst følger reelle forsøg på udnyttelse af de (potentielle) sårbarheder, som er identificeret. De ovennævnte værktøjer har righoldige muligheder for dette, mens de mere specialiserede angreb kræver viden og snedighed hos angriberen.
Udnyttelsen kan ske i form at direkte brud på kryptering, generelle sårbarheder (sådan nogle som kan slås på i CVE), "forkerte" inputs eller på anden vis invalidering af antagelser, som designerne har haft omkring funktionen af koden eller systemet.

Afrapportering Herefter følger en afrapportering, som regel inkluderende anbefalinger til udbedring af sårbarhederne, til kunden. Det kan f.eks. ske i form af en henvisning til en sårbarhedsdatabase og et proof-of-concept, som viser den konkrete udnyttelse af sårbarheden i systemet.
Herefter vil pen-testeren eller producenten typisk selv implementere løsninger på sårbarhederne, som kan reducere eller eliminere disse.

Opsummering Undersøgelser vha. ovenstående metode afhænger især af brugen af automatiserede værktøjer; for de "mindre vidende" angribere, fordi de ikke kan andet og for de "klogere" pga. muligheden for at automatisere (og derved spare tid) og ramme flere systemer

[34] http://nmap.org/. Lignende værktøjer er eksempelvis Nessus, Metasploit, NetStumbler og Qualys

på en gang. Ved en ikke-direkte adgang til produktet, udgør det blot en lille dråbe i det store internet-hav, for servere bombarderes konstant med "urene" pakker og i dette, vil en beskyttelse mod kendte og udbredte sårbarheder, være den klart bedste at investere tid og penge i for en virksomhed.
Imidlertid er der også mere direkte, specialiserede angreb, hvor vi snakker om sikkerheden "indeni" produktet – altså omkring funktionerne. Her er de automatiserede værktøjer mindre behjælpelige og der stilles større krav til angriberens viden.
Her kunne en ondsindet angriber vælge at erhverve en model af produktet og systematisk undersøge det for sårbarheder. I dette tilfælde ville man på samme måde opsøge informationer om systemet og hvilke komponenter det inkorporerer (f.eks. kendte protokoller), men nu øges truslen væsentligt mod de rent fysiske inputs og inputs med ukendte, proprietære protokoller, eksklusive for produktet. Fremgangsmåden vil dog stadig blive afgjort af hans eksisterende viden, hvorfor en modellering, hvor udbredelsen af inputtet tillægges en vis værdi, stadig giver mening, fordi det iht. ovenstående direkte (i de fleste tilfælde) kan øge angriberens chancer for at identificere en sårbarhed. At afhænge af det er dog ikke tilstrækkeligt og derfor baseres vores score heldigvis ikke kun på dette.
Den eneste måde at hindre dette på, er at udforme programmet korrekt, men så er vi ovre i principperne for robust programmering [3] eller generelle datasikkerhedsforanstaltninger, som angivet under afsnit A.2.5.1.

Ville man som producent forsøge at få identificeret truslerne, ville man altså bede en pen-tester om at udføre de samme trin som en ondsindet hacker ville gøre og så krydse ben og arme for, at han er tilstrækkelig dygtig.
Vores metodik kan dermed (præcis som de præsenterede, lignende modeller) bruges som et billigt og effektivt værktøj til at undersøge, hvor problemerne potentielt vil være at finde for design- eller udviklingsafdelingen og give dem idéer til videre handlinger, som kan løse disse. Desuden kan det senere hen bruges som et værktøj ifm. pen-testing processen til at identificere sårbarheder jf. ovenstående.

6.2.1 AccessZone

Forfatter: Daniel

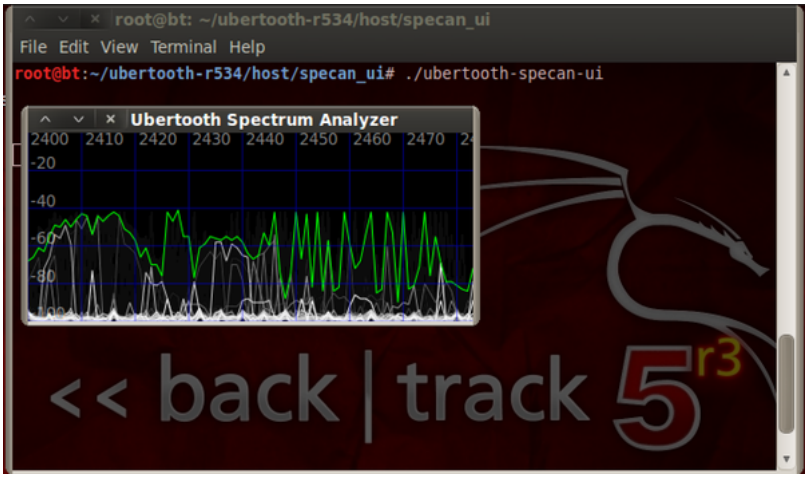
Efter vejledningens modellering er udarbejdet for AccessZone-dørlåsen, kan vi udlede, at den bedste måde at forsøge at hacke dørlåsen på, er ved at kompromittere bluetoothdelen, som beskrevet i afsnit 6.1.1.
Ved informationsindsamling omkring AccessZone-systemet fra databladet og hjemmesiden, finder vi bl.a., at der findes en kritisk *master PIN-code*. Denne master PIN bruges til al administration af systemet, så hvis den kan findes, kan vi ved brug af eget bluetoothudstyr (f.eks. en

mobiltelefon), sende kommandoer til låsen. Et eksempel på en række problematiske kommandoer vi kan sende, er først sletning af brugerdatabasen og dernæst en ændring af master PIN'en. Hvis det lykkes, kan de autoriserede brugere af systemet ikke tilgå eller anvende det længere; det vil formentlig kræve en fysisk genstart af systemet via serviceindgangen for at kunne bruge det igen — en operation som kræver professionel assistance fra MVC-data.

6.2.1.1 Opsætning og udstyr For at kompromittere sikkerheden skal vi altså bruge en metode til opsnapning/læsning af bluetoothtrafik. Her er valget faldet på en Ubertooth-enhed, som er et open source-projekt til lige akkurat bluetooth sniffing[35]. Med hardwaren følger også en del software, som på linux-platforme kan bruges sammen med Kismet[36] og opsnappe og læse bluetoothpakker.

Vi har valgt at sætte vores system op med Backtrack 5 som operativsystem. Backtrack er et linuxbaseret operativsystem, som med sloganet "*The quieter you become, the more you are able to listen*", fokuserer på sikkerhedsanalyse (til dels også den mere uofficielle del af området). Fordelen ved dette er, at der med Backtrack 5 følger en del værktøjer der bruges til sikkerhedsanalyser, f.eks. som Kismet og Wireshark og det samtidig ikke er den store udfordring at få plugins til f.eks. bluetooth-sniffing. En af udfordringerne med Backtrack er dog, at udviklingen er stoppet[37] og det er derfor ikke ligetil at finde drivere og programmer, som kan eksekveres på systemet.

Opsætningen af Ubertooth-enheden foregår i to dele: I første omgang skal vi kontrollere, at enheden virker og at der kan måles trafik. Dette gøres ved at bruge den indbyggede *Spectrum Analyzer*. Resultatet af dette kan ses på figur 11.



Figur 11: Backtrack med Ubertooth Spectrum Analyzer

[35] http://ubertooth.sourceforge.net/
[36] www.kismetwireless.org
[37] Det er genopstået som Kali Linux: http://www.kali.org/

Spectrum Analyzeren testes ved at sende tilfældige bluetoothpakker mellem to bluetoothenheder (i vores tilfælde mobiltelefoner) og vi ser, at det giver et udslag.
Dernæst skal vi have sat udstyret op til også at kunne læse pakker. Dette er en væsentligt større operation, men heldigvis er Backtrack udstyret med de fleste af værktøjerne. Vi bruger et plugin til netværksanalyseprogrammet Kismet til at opsnappe trafikken mellem vores bluetoothenheder. Dernæst åbnes logfilerne fra Kismet i Wireshark[38], da dette kan bruges til en mere dybdegående analyse af trafikken. Når det er sket, er det tid til at prøve at analysere trafikken mellem AccessZone-systemet og en mobiltelefon.

6.2.1.2 Resultater Det er lykkedes at opsnappe trafikken mellem AccessZone-systemet og en mobiltelefon som sender kommandoer. Trafikken er umiddelbart krypteret med mere end standard bluetoothkryptering, hvilket også er tilrådeligt da der allerede tilbage i 2001 blev sat spørgsmålstegn ved bluetooth's standardkryptering [12].
Det er i skrivende stund ikke lykkedes os at bryde krypteringen af dataoverførslen mellem dørlåsen og mobiltelefonen, hvilket er en nødvendighed for at kompromittere sikkerheden i systemet.
En kryptoanalyse af bluetoothpakkerne ville i første omgang skulle identificere hvilken form for kryptering der er anvendt. Når dette er gjort, vil det være muligt at lave et såkaldt *known-plaintext* angreb, hvor vi kender klarteksten vi sender, og kan opsnappe den krypterede tekst. På den måde vil det være muligvis kunne lade sig gøre at bryde krypteringen, men kryptoanalysen ligger ud over projektets scope. Vi kan altså notere os, at det er muligt at opsnappe trafikken mellem AccessZone og administratormobiltelefonen, men at en kryptoanalyse er nødvendig for at bryde sikkerheden.

6.2.1.3 Evaluering Ved brug af vejledningens metode, blev bluetoothinputtet angivet som det mest kritiske input på AccessZone-udstyret. Dette var at forvente fra start, så det var ikke den store overraskelse. MVC-data har dog taget hånd om det kritiske ved bluetooth og lagt kryptering på datastrømmen mellem mobiltelefonen og dørlåsen. En yderligere sikkerhedsevaluering af dørlåsen vil kræve en grundig kryptoanalyse af datastrømmen, men projektets omfang forhindrer dette i at blive gjort nu.
Dette betyder også at bluetoothinputtet måske ikke er så kritisk som vores modellering har påvist, men dette kan begrundes i, at vi i vores modellering ikke har haft tilstrækkelig kendskab til produktet til at kunne sætte nøjagtig score på den pålagte sikkerhed i trin 4.3, jf. afsnit A.3.4. Hvis modelleringen havde været udført af en systemdesigner fra MVC-data, havde udfaldet muligvis set anderledes ud, så det er en faktor vi er nødt til at tage højde for, når vi holder ovenstående op imod vores modellerings resultat. Vi er dog stadig overbeviste om, at den stør-

[38] www.wireshark.org - Wireshark er et meget populært værktøj for både administratorer og sikkerhedstestere og tilbyder righoldig funktionalitet for analyse af (mange) datapakker

ste sandsynlige sårbarhed ligger på bluetooth-inputtet. Det er formentlig også her en angriber ville fokusere sit angreb, fordi automatiserede værktøjer kan øge hans succesrate ift. hvis han fokuserede sine kræfter på strømpins'ne, hvor han i øvrigt ville udsætte sig selv for en vis risiko, når selve hacket skulle udføres.

6.2.2 Smart TV

Forfatter: Rasmus

Når vi vender modelleringen af Smart TV'et på hovedet, har den givet os nogle mulige indgangsvinkler for et angreb. Da vi ikke har haft TV'et fysisk, er det svært at teste systemet i praksis, men vi vil alligevel i det følgende se på tre konkrete angreb, som vil kunne kompromittere TV'et.

6.2.2.1 Sniffing af datapakker I november 2013 fik mange mennesker med et LG Smart TV et chok. Det viste sig, at LG, på alle deres TV, indsamlede data om hver kanal der blev set og hvilke mediefiler (også lokale fra en USB), som blev afspillet[39]!
Dette blev opdaget af "DoctorBeet" efter, at han havde set en reklame på TV'ets velkomstskærm og ville undersøge dens "smarte" natur nærmere jf. et *Smart Ad*-program, som LG kørte.
Ved direkte at læse pakker på sit lokale netværk vha. Wireshark, fandt han pakker sendt over HTTP fra sit TV til et LG-domæne, som bl.a. indeholdt kanalnavn i klartekst.

Intentionen har været god nok fra LGs side, idet man har kunnet servere målrettede reklamer direkte til forbrugeren på forsiden af deres velkomstskærm, men det giver nogle image- og troværdighedsproblemer for dem. Dette først og fremmest fordi forbrugere ikke kan lide at blive udnyttet til at tjene penge på (i hvert fald ikke, hvis de ikke får en synlig belønning tilbage – se bare Facebook), men samtidig også fordi man snager i deres private data og især fordi funktionen til at slå denne indsamling fra, bare er til pynt!
To ting ville have gjort denne sag bedre for LG, end blot at henvise til deres alenlange brugerbetingelser: At indstillingen til at slå det fra havde virket og at man havde krypteret data'en, for ikke nok med, at de indsamler denne information, så behandler de den også ganske lemfældigt i transit.

Wireshark ville også kunne bruges til at opsnappe passwords eller login-cookies sendt på åbne netværk, eller netværk, hvor angriberen har opnået adgang, som i næste afsnit.

6.2.2.2 WiFi-cracking Ovenstående logning og megen anden data som flyder over den trådløse forbindelse mellem TV'et og mange andre enheder i vores målgruppe, kan være yderst

[39]Den originale afsløring: http://doctorbeet.blogspot.co.uk/2013/11/lg-smart-tvs-logging-usb-filenames-and.html

interessante for en angriber.
Samsung Smart TV'et tilbyder derfor både WEP, WPA-PSK og WPA2-PSK (med WEP, TKIP og AES som krypteringer) og desuden mulighed for WPS. Det sker givetvis for bagudkompatibilitet med kundernes udstyr, men desværre er disse godkendelsesprotokoller sårbare og kan brydes på kort tid.

Tilgangen mod dem er forskellige. En ældre og meget usikker protokol som WEP, har så mange sårbarheder i sin opsætning, at den kan brydes på relativ kort tid[40]. Den bør man om ikke andet slå fra, så sætte indtil flere advarselsbokse op, inden det kan aktiveres.
WPA og WPA2 med en *pre-shared key*, hvilket er typisk i private netværk uden central key management i form af f.eks. AAA-servere med RADIUS, er, som følge af computeres altid stigende ydeevne, under pres, fordi den statiske hovednøgle kan brydes ud fra handshaket, når en klient authentificeres. Afhængigt af nøglens kompleksitet og angriberens ordbog, kan nøglen findes indenfor en overskuelig tid[41].
Kører netværkets AP'er med WPS aktiveret (og det er der stor sandsynlighed for i mange private hjem), er de endvidere udsatte for, at en angriber kan brute-force sig til den korte, statiske WPS-kode og subsidiært få nøglen til netværket og dermed opsnappe trafikken mellem systemet og producenten eller andre servere[42].

6.2.2.3 IR-fuzzing med USB-UIRT og Girder Et sidste relevant angreb på TV'et, kan ske via dets infrarøde modtager. Infrarøde sendere er meget udbredte i hjemmeelektronik, fordi det er nemt og billigt at implementere, idet det blot består af en serie hurtige blink fra en diode i det infrarøde spektrum. Der findes bl.a. enheden USB-UIRT[43] (se figur 12), som er en USB-forbundet **U**niversal **I**nfrarød **R**ecevier og **T**ransmitter, som kan bruges sammen med automatiseringsprogrammer (f.eks. Girder[44]) eller direkte som enhed i f.eks. Java og C#.
Dermed haves pakken til at udføre et fuzzing-angreb[45]. Fuzzing er et generelt udtryk for en black box-test[46] af inputs, som involverer at (automatisk) sende mere eller mindre malformaterede inputs mod en enhed for at afsløre fejl i håndteringen af disse i controlleren.

Den specifikke udførsel med denne pakke, ville være at undersøge to sider: Hvordan systemet håndterer helt tilfældige signaler og hvordan det håndterer mange, højfrekvente, korrekte inputs. Årsagen til dette valg, er, at systemet ikke er sårbart overfor den første type, hvis det

[40] http://lifehacker.com/5305094/how-to-crack-a-wi-fi-networks-wep-password-with-backtrack
[41] http://www.aircrack-ng.org/doku.php?id=cracking_wpa
[42] http://lifehacker.com/5873407/how-to-crack-a-wi-fi-networks-wpa-password-with-reaver
[43] http://www.usbuirt.com/overview.htm
[44] http://www.promixis.com/girder.php
[45] http://en.wikipedia.org/wiki/Fuzz_testing
[46] En black box-test er en softwaretestmetode, som undersøger koden uden at se på den specifikke opbygning af denne. Se http://en.wikipedia.org/wiki/Black-box_testing

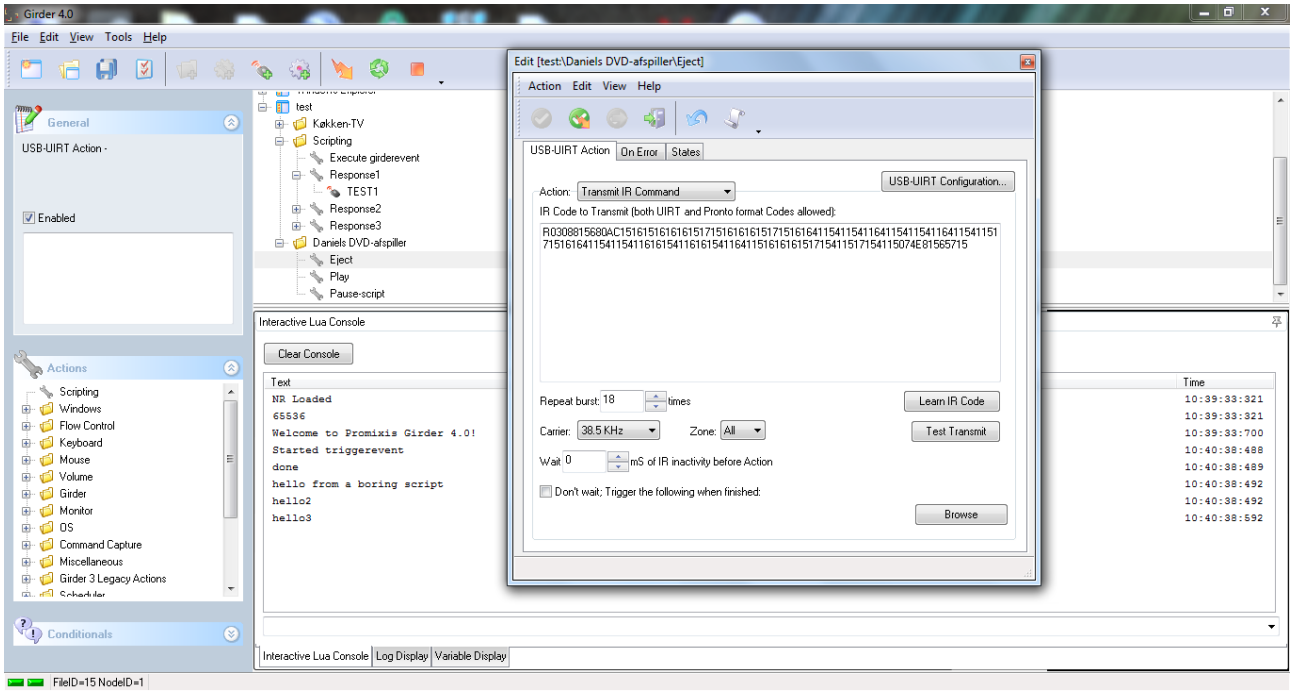
Figur 12: En USB-UIRT

på korrekt vis kan kassere forkerte inputs uden at lade sin hukommelse fylde eller på anden vis bryde sammen. Med det andet angreb kan vi så undersøge, om vi evt. kan fylde en stak i selve controlleren eller måske dybere nede i systemet, hvis vi med høj frekvens beder om en kanalændring, ændring af indstilling osv.

Grundet uforudset sygdom, har det ikke været muligt at få afprøvet mulighederne med at programmere USB-UIRT vha. dets Java-implementation (for øget kontrol med sendehastighed mv.). I Girder har man dog mulighed for at optage, redigere, gemme og afspille IR-koder med et bestemt interval (se figur 13), så det kan til dels træde i stedet for. Det er en god start at indsamle information om produktet først. I denne kontekst betyder det, at vi bl.a. har set på hvordan Samsungs IR remote controller (fjernbetjeningen) fungerer. Det hjælper os til fra starten, at se bort fra muligheden for at sende helt malformaterede koder til enheden, fordi den behandler dem som en state machine, så et input straks kasseres, hvis det ikke er gyldigt [24]. Forsøget skal altså ske med korrekte inputs, men sendt på en utilsigtet måde (dvs. en måde, som fjernbetjeningen ikke kunne have gjort).
Udover dette er IR-koderne i HEX-format og ikke umiddelbart forudsigelige i deres udformning[47] – specielt ikke dem vi optager fra fjernbetjeningen, fordi dette oftest er *toggle commands*[48] og de derfor udfører flere funktioner. Vi kan dog sagtens bruge både de optagede og dem fra lister af diskrete IR-koder, for uanset hvad, opfylder vi formålet med fuzzing.

Hvad der reelt ville komme ud af det, er svært at sige. Forhåbentlig har man brugt en veludformet IR-chip/-controller, hvor ovenstående slet ingen indflydelse har eller måske er der nogle

[47] http://www.remotecentral.com/cgi-bin/codes/samsung/tv_functions/ og der findes ingen dokumentation på, at state machinen fungerer på en, for udenforstående, logisk måde
[48] http://www.engadget.com/2009/02/05/hd-101-discrete-ir-codes/



Figur 13: Interfacet i Girder efter optagelsen af en IR-kode fra en fjernbetjening

helt specifikke omstændigheder, hvor man kan bringe TV'et til at reagere utilsigtet (især i lyset af de mange mulige kombinationer, som et så featurerigt produkt har).

Figur 14: En USB-UIRT sender IR-signal til en DVD-afspiller. Den venstre, røde diode sender i spektret for synligt lys, mens de to lilla er infrarøde dioder, som transmitterer signalet til afspilleren

Vi har specifikt afprøvet dette i mindre udstrækning på et fjernsyn af ældre dato og en DVD-afspiller (på figur 14).
På fjernsynet kunne vi få volumen til at stige konstant – også i tiden efter transmitteringen var stoppet. Det stoppede helt med at reagere på inputs (både IR og fysiske knapper) og skruede

blot helt op. Det kan være, at dens kommandostak blev fyldt, men så burde det være vendt tilbage til normal funktion efter nogen tid.
På DVD-maskinen observerede vi, at kommandoerne havde flere betydninger eller den simpelthen opgav at udføre dem. I hvert fald kunne vi få maskinen til gentagende gange at åbne og lukke DVD-skuffen (kun begrænset i hastigheden af motoren) og andre gange ville den slet ikke reagere. Den ville kun acceptere et ulige antal signaler, når dens skuffe var lukket og åbnede altid kun én gang, mens den reagerede på de fleste sendinger, når den var åben – uafhængigt af antal, men uden korrelation mellem det antal gange signalet blev gentaget og hvad der reelt skete.

6.2.2.4 Evaluering Andre angreb end de ovenstående eksisterer også[49], men detaljerne for dette er ikke tilgængelige.

Holder vi resultatet af ovenstående mulige angreb op mod modelleringen, skal vi kunne se en sammenhæng i farvningen af inputs kontra de angreb, vi antager er mulige.
På figur 22b ser vi, at det infrarøde input er farvet orange, altså det næsthøjeste niveau. WiFi og ethernet er derimod begge farvet lysegrønne, altså næstlaveste niveau.
Det passer faktisk udmærket med ovenstående! Fuzzing-angrebet mod IR er ikke det eneste mod dette input og det er helt sikkert et sted, hvor man ikke har tænkt sikkerheden særlig højt. I praksis kan det dog vise sig, at der er lavet en robust implementering af IR-controlleren, i hvilket tilfælde Samsung heldigvis vil kunne afvise den sandsynlige sårbarhed, som vores modellering ellers viser dem.
Hvad angår angrebet på WiFi'en, så er angrebet generisk og kunne som sådan berøre alle systemer i vores målgruppe. Som producent vil Samsung ikke kunne gøre meget yderligere, fordi de er nødt til at følge standarderne, men må sætte deres lid til at 802.11i-arbejdsgruppen eller andre relevante enheder udkommer med nye tiltag. Selve angrebet mod datatrafikken, er dog en specifik sårbarhed, som er indeholdt i netværkscontrolleren. Som beskrevet, er det ikke fremhævet særligt i vores metodik, men i afsnit A.2.5.1, står der klare modtræk, som ville kunne afhjælpe dette specifikke problem. Samtidig stiller vi i afsnit A.2.4.1 spørgsmål til overvejelse og selvransagelse omkring de indre services og deres betydning for brugernes data, virksomhedens renomé mv., som kunne have ledt dem i retning af denne "sårbarhed".

Efter denne teoretiske udførsel af vejledningen på Smart TV'et og den praktiske på dørlåsen, har vi set to eksempler på, at metodikken reelt er anvendelig på et par produkter i målgruppen. Samtidig har vi med de to ovenstående eksempler på forskellige angreb også set, at den modellering vi har produceret, rent faktisk vil have kunnet finde anvendelse til at imødekomme

[49] http://arstechnica.com/security/2012/12/how-an-internet-connected-samsung-tv-can-spill-your-deepest-secrets/

disse angreb, idet modellen viste, at bl.a. disse inputs var mere sårbare end de fleste andre i systemerne.
Specielt helt generiske angreb mod systemerne, vil den være god til at påvise, idet dette er medtaget i udformningen, men også funktioner, som ikke nødvendigvis er "populære", men har kritisk funktion eller indhold, vil blive tillagt en ekstra værdi i modelleringen og derfor øget opmærksomhed.

7 Diskussion

Forfatter: Daniel

Metodikken som helhed er fyldestgørende (og endelig), men alligevel er der flere små forbedringer og overvejelser, som er dukket op undervejs og som kan være interessante at arbejde videre med. I det følgende gennemgår vi en række interessante forbedringer eller kritiske dele af vejledningen, som vil kunne ændres i en ny iteration af denne.

Projiceringen af konsekvensen af en funktion og ud på inputs, har vi været i tvivl om hvorvidt, er udført på en optimal måde. Fordelen ved det nuværende ligger i, at vi kan propergere "problemet" fra alle funktionerne ud på perimeteren af produktet og derved samle det ét sted. Imidlertid får vi nu svært ved at vurdere sikkerheden omkring selve funktionen. Det kan være, at der er yderligere databeskyttelse her, men det kan ikke pålægges et input. Skulle det modelleres med den nuværende opsætning, ville man være nødsaget til at lave separate enheder af en funktion og hvert af dets inputs og her vurdere sikkerheden isoleret.
Vi har dog opnået en sammenlægning af elementerne, som man ikke har i andre modeller, hvor man nedbryder systemet i enkeltdele og *udelukkende* behandler dem som sådan. Som beskrevet i teorien, kan det betyde skjulte sårbarheder, når systemets delelementer kombineres til slut, hvorfor vores sammenlægning kan give grundlag for en mere præcis modellering.

I relation til ovenstående, har vi undladt at medtage sikkerhedsforanstaltninger på services og funktioner i vejledningen. Da en del af disse formentlig har indbygget en eller anden form for sikkerhed i forvejen pr. design, vil det være mere fyldestgørende for vejledningen af tage disse med for at give det komplette billede.
Det kræver imidlertid, at man har et større kendskab til produktet og kommunikerer med tredjepartsudviklere, da det kan være svært at vurdere sikkerheden på apps. En ny iteration af vejledningen kan inkludere en form for sikkerhedsanalyse af funktioner og services, når der laves risikoanalyse i trin 4.
Samtidig skal det dog påpeges, at vores metode er rettet mod en samlet hardware-/softwarekombination, hvor funktioner/services og inputs hænger uløseligt sammen, hvorfor den manglende modellering indeni systemet for så vidt er udenfor metodikkens anvendelsesområde! Hvis man er interesseret i at modellere sårbarhederne mod softwaren, har man netop en anden metode som Microsofts STRIDE eller en variant heraf, som gør det bedre og arbejder specifik med dette.

Der kan i nogle produkter findes forbindelser mellem services og inputs, som eksisterer rent fysisk, men som er låst på den eller anden måde ved hjælp af et stykke software (á la `lockConnect==true`). Denne form for beskyttelse kunne være udsat for en hacker og det kunne være en stor sikkerhedsrisiko, hvis det eksisterer. Derfor kunne det være en idé at nævne denne problemstilling i

trin 5.
Vi har dog ikke nogen henvisninger, som peger på, at det er en udbredt metode. Samtidig vil en teoretisk kompromittering af dette, skulle gå igennem en anden del af produktet, f.eks. den centrale controller, hvorved den bør være indfanget som normalt; derfor har vi valgt ikke at medtage det i vejledningen.

Den UML-lignende models opbygning, er lavet på en enkel og logisk måde, men vil kunne simplificeres i en ny iteration af vejledningen. Den øverste del af kassen kunne gøres mere overskuelig, hvis man fjerner "controller"-kasserne. Størstedelen af dem er bare en boks på linjen mellem inputtet og funktionen; de deler sjældent hverken input eller funktion, som var den oprindelige tanke. I de få tilfælde hvor de gør, kan man blot føre lige linjer til alle funktioner.
I samme kategori ligger de uafhængige komponenter vi har med (som delt hukommelse og RAM). De har ikke vist sig at have den betydning i modellen, som vi havde regnet med og derfor kan det overvejes at udskrive dem. Vi har beholdt dem i denne iteration, fordi de er med til at markere trinnet mellem funktioner og controllers, hvor den centrale styring er placeret for nogle systemtyper og i alle tilfælde vil eksistere.

Forholdet mellem funktioner og inputs i vores metodik er kun bygget op om hvilke forbindelser, der ligger imellem dem.
Vi har diskuteret om det er for stor en antagelse at arbejde med. F.eks. har TV'ets infrarøde input adgang til de fleste funktioner og er dermed et temmeligt kritisk input. Hvis IR-controlleren imidlertid kun accepterer de signaler som den officielle fjernbetjening kan afsende, kan det være ligegyldigt hvor meget controlleren har forbindelse til, da den vil ignorere alle forespørgsler som fjernbetjeningen ikke kender. Dette vil medføre at fuzzing ikke vil have nogen effekt og inputtet dermed er mindre kritisk end antaget.
I en ny iteration af vejledningen kan forholdet mellem controllers og funktioner i produktet tages op til overvejelse, således, at "begrænsede" inputs sikkerhedsmæssigt ikke bliver tillagt en så kritisk score, som de gør i den nuværende udgave af vejledningen. Forudsætningen for dette er, at disse sikkerhedsforanstaltninger er robuste således, at controlleren ikke bryder sammen ved ondsindet input (som beskrevet i 6.2.2) og at man ydermere har kendskab til disse sikkerhedsforanstaltninger.

Opbygningen af trin 5 og hvordan modtræk præsenteres, har været oppe og vende et par gange. Optimalt set ville der være en lang liste med forskellige inputs og protokoller med tilhørende angrebs- og sikkerhedsforanstaltninger, hvilket vi arbejdede ud fra i starten. Dette ville dog være en *kæmpe* opgave, gå langt ud over projektets omfang og blot være en endeløs indtast-

ning fra diverse referencer (obskure hjemmeside, CVE-databasen, security bulletins[50] etc.). Det ville samtidig betyde, at vejledningen ville blive forældet ligeså snart en kendt protokol bliver kompromitteret eller der kommer nye standarder for inputs – og så er den metodiske tilgang i vejledningen også forsvundet.
En måde at begrænse dette på, ville være at lave henvisninger til eksternt materiale. Det ville dog igen kræve en lang liste med inputs og protokoller, som i virkeligheden aldrig ville blive komplet og vejledningen ville igen forholdsvist hurtigt blive forældet.
Der har i de eksisterende metoder ikke været meget inspiration til hvordan dette kunne gøres, så vi har valgt at lave en generel udlægning af forbedringer ud fra trusselstyper (af de årsager diskuteret i 4.2.4) og så lægge op til, at man som producent henter viden udefra, hvis ikke man har de ressourcer der skal til for at arbejde videre med dette.

Endeligt har modelleringen af TV været påvirket af, at vi ikke har haft et fysisk TV til rådighed. Samsung har offentliggjort nogle tekniske manualer, men disse er ikke omfattende nok til, at det er muligt for os at lave en fyldestgørende analyse og modellering af fjernsynet. Derfor er denne blevet mere teoretisk baseret end ellers og det resulterer i, at visse dele af produktet kan være overset og at nogle ting kan argumenteres at burde være tillagt større opmærksomhed, end vi har gjort.
Metoden giver dog stadig et godt billede af, hvordan man ville have udført evalueringen i praksis og på et mere kompliceret produkt, så dens værdi for vejledningen er uomtvistelig.

7.1 Scoren i risikoanalysen

Skalaen, som er brugt til scoren i sikkerhedsvurderingen, er generelt noget mindre end ved de sikkerhedsmodeller, som er beskrevet i afsnit 3.3. Det betyder, at den samlede risikoscore for produktets input ikke har så stor en skala at spænde over og inputs, som i virkeligheden ikke ligger samme sted på skalaen, kan komme til at ligge op og ned ad hinanden. Dette kan have betydning for prioriteringen af inputs i trin 5, hvilken igen er et problem for vejledningens resultats korrekthed i sidste ende.
På den anden side ville en større spændvidde i skalaen betyde, at mere af det samlede resultat var lagt over på subjektive vurderinger af risiko, trussel og sikkerhed og derved kunne resultatere i, at modelleringen ville blive væsentlig forskellig alt efter hvem der udfører analysen. Vi har derfor, for at holde evalueringerne så konsistente som muligt, holdt os til mindre skalaer, hvor vi kan sætte en etiket på hvert trin jf. afsnit 4.2.4.
Den mindre skala betyder også, at input helt uden sikkerhed ikke bliver "straffet nok" i modellen. I en ny iteration af vejledningen bør man forsøge at tage højde for dette, så scoren bliver

[50]Mange sikkerhedsfirmaer og andre store softwarefirmaer publicerer nyhedsbreve om sårbarheder og forbedringer i deres produkter

udvidet, men det skal ske uden, at det konsistente billede vi har i den nuværende udgave, bliver fejet væk. En metode kunne være at bruge et system, som ligner DREAD mere (se afsnit 3.3.1). Her vurderes på flere og smallere parametre og dernæst tages et gennemsnit (kritikpunktet er dog netop, at disse nye parametre er for specifikke og svære at vurdere for ansatte uden tilstrækkelig datasikkerhedsteknisk viden).

Scoren i trin 4.2 er på sin vis hængt op på hvor "kendte" de forskellige inputs er. Det er selvfølgelig ikke tilrådeligt fuldstændigt at sætte sin lid til *security through obscurity*[51], men det er et faktum, at udbredte protokoller, som findes i mange produkter, i langt højere grad er interessante for angribere at gå i lag med, blandt andet fordi der findes et bredt udvalg af automatiserede værktøjer og informationer om disse er så lettilgængelige. Har man imidlertid et tilstrækkelig interessant eller værdifuldt (i hackermæssig forstand) produkt, vil dette imidlertid kun være en mindre hindring for den dygtige hacker, hvorved kompromittering kun vil være et spørgsmål om tid.
Det er derfor ikke sikkert, at denne vurdering er tilstrækkelig god, men vi synes dog, at den giver et godt billede af den trussel, systemet potentielt udsættes for i sit miljø, hvilket netop er hvad vi vil. Selve konsekvensen (og derved den potentielle værdi af systemet) vurderes i et selvstændigt trin, så på den måde er begge sider dækket.

Vi har haft en del muligheder oppe at vende når det kom til hvordan man skulle give score i trin 4.3. Vores nuværende løsning med at give en score baseret på ingen, middel eller høj sikkerhed og så dividere den foregående score med dette tal, virker umiddelbart efter hensigten og går i tråd med vores målsætning om, at karaktergivningen skal være konsistent.
Kigger man på distributionen af alle de mulige risikoscores efter trin 4.3, får man dog et lidt andet billede. Figur 25 (i afsnit B) viser denne distribution og man bemærker hurtigt, at der er en væsentlig overvægt af lave værdier. Dette vil i første omgang medføre, at den endelige modellering kommer til at fremstå med en lavere sårbarhed, end hvad tilfældet i virkeligheden er. Billedet vil dog være konsistent over alle modelleringer.
Vi har forsøgt at opveje dette, ved blandt andet at give den mest kristiske farve i modelleringen et større spændvidde. Man kommer dog ikke udenom, at sandsynligheden for at et input scorer lavt, er stor.
Et alternativ til dette er at bruge samme skala i trin 4.3, men at vende skalaen om og multiplicere den på den foregående score i stedet for at dividere[52]. Distributionen i dette tilfælde kan findes på figur 26 og man bemærker at der er sket en *lille* forbedring i forhold til den forrige,

[51] Et princip om at sikkerheden i ens kode/protokol o.lign., baseres på offentlighedens manglende kendskab til denne; http://en.wikipedia.org/wiki/Security_through_obscurity

[52] Dette er en typisk fremgangsmåde for risikomodellering indenfor maskinteknologi. Scoren for konsekvens, trussel og sikkerhed multipliceres, så det ville vi undersøge

men vi har dog stadig en stor overvægt af lave værdier. Da medianen på de to distributioner er ens, har vi valgt at holde os til den oprindelige.
En ændring af distributionen ville kræve, at man revurderede hele skalasystemet og vægtede anderledes (ved at opstille nye kriterier for hvert trin). Det ville betyde, at den konsistens vi gerne vil have i vejledningen forsvinder og vi har derfor valgt, at holde os til lav-middel-høj scoren og i stedet korrigere på en simpel måde.

8 Konklusion

Forfatter: Rasmus

Dette projekt har haft til formål at udvikle en abstrakt metodik og en vejledning i denne, som kan kortlægge sårbarheder i et stykke moderne forbrugerelektronik. I afsnit 2.2 oplistes succeskriterierne for projektet, hvor målet er at få lavet en metodik, som tillader elektronikproducenter at analysere sikkerheden i deres produkter. Dette indebærer at identificere alle angrebsflader ("huller i rustningen"), hvor en angriber kan komme i kontakt med systemet, identificere de sikkerhedsmæssigt kritiske dele af systemet og endeligt give vejledning til at øge sikkerheden. Vi har gennem teori baseret på andre modeller og grundlæggende datasikkerhedsfaglig viden, analyseret os frem til indholdet af en samlet vejledning, som opfylder netop ovenstående. Vores mål har været at gå efter at invalidere antagelser ved helt objektivt at modellere produktet og dermed at afdække dets eksisterende sårbarheder bedst muligt – uden påvirkning af hvad producenten selv måtte mene.

Vores metodik er et systematisk værktøj med en fuldstændig tilgang til at afdække potentielle sårbarheder i et stykke moderne forbrugerelektronik, som kan bruges undervejs i udviklingsprocessen hos firmaer, hvor pengene er små. Den kan bruges i stedet for en usystematisk tilgang, hvor man kun kontrollerer specifikke dele af systemet og kan ydermere danne udgangspunkt for en pen-testers videre arbejde og derved spare noget tid på indsamlingen af informationer om systemet. Som output har den en gradueret, farvet model, hvorfra resultaterne direkte og nemt kan aflæses af alle.
Den opfylder ydermere kravene, fordi den inkorporerer kendte elementer fra fagområdet i form af visuelle fremstillinger (UML og heat maps) og modeller (risikomanagement, rapportopbygning), som gør brugen mere ligetil og hurtigere. Endvidere motiveres hvert trin, sådan at man forstår essensen og vigtigheden af hvert af disse.
Endeligt gives der konkrete råd til modtræk ud fra en heldækkende række af sårbarhedstyper på de forskellige komponenttyper i systemerne, så producenten har et udgangspunkt og nogle konkrete løsningsforslag, som der kan arbejdes videre med.

I tilgift til selve vejledningen, har vi understøttet metodikken med to gode eksempler i hver sin

ende af spektret af både kompleksitet og funktion, som viser anvendelsen af både metoden og den dertilhørende modellering.
Vi har desuden kigget på en række angreb, som kan udføres imod disse to systemer, for på den måde at se, hvorvidt vejledningen afspejler den virkelighed, som produktet agerer i.
Vi fandt for alle fire angrebtypers vedkommende ud af, at modelleringen også viste disse sårbarheder, sådan, at det ville være sandsynligt, at producenten med resultatet i hånden og en person med tilstrækkelig praktisk viden om sikkerhed i sådanne systemer, vil have kunnet identificere disse sårbarheder og implementere foranstaltninger.

I diskussionen vender vi flere afspekter omkring vejledningen og metodikken.
Vores metode er rettet mod en samlet hardware-/softwarekombination, hvor funktioner/services og inputs hænger uløseligt sammen. Derfor er den mindre god, hvis man er interesseret i at modellere sårbarhederne mod software; her har man dog andre metoder (som Microsofts STRIDE).
Desuden har vores vejledning ikke fokus på at løse de direkte trusler, men i stedet på hvor de potentielle sårbarheder findes. Det er den normale tilgang for de gængse modeller og det tolkes som den mest effektive måde for en producent, som vil se på hvor vedkommendes produkt har kritiske sider. En hacker vil i mange tilfælde blot lade sig nøjes med én af disse sårbarheder og derfor er det vigtigt, at man på en systematisk måde får evalueret hele produktet, så alle sårbarheder fremhæves.

Vi har identificeret mange sårbarheder på de to produkter – og de fleste sandsynlige, men målgruppen er bred, angrebsfladerne virkelig brede og vores eksempler blot to ud af mange, så vi kan ikke sige, ud fra de præsenterede resultater, om vi har dækket fuldstændigt ind. En vigtig del af diskussionen går på den score, som benyttes i vores eksempelmodellering i trin 4 (afsnit A.2.4 og A.3.4) til at prioritere sårbarhederne i sidste ende. Den er lavet ud fra en del antagelser, fordi vi kender ikke produkterne "indefra" som en producent gør. Især med TV'et har vi måtte gætte på, hvordan tingene ser ud inde bag plastikken. Det betyder, at den endelige sikkerhedsvurdering på hvert input, ikke nødvendigvis ville være den samme, hvis det var en designer fra Samsung eller MVC-data, som havde lavet modelleringen, men det er den betingelse vi har arbejdet under.
De vurderingsspørgsmål som ligger til grund for scoren i trin 4, er netop udformet til, i videst muligt omfang, at minimere de antagelser analytikeren er nødt til at gøre (for på den måde, at minimere subjektive holdninger til systemet), men selve implementeringen af dette scoresystem, kan muligvis, vurderer vi, udformes endnu bedre i en fremtidig udgave af metoden. Målet kunne være i endnu højere grad at fremme de objektive vurderinger, men vi konkluderer dog efter diskussionen af dette, at vi er nået i mål også på den front.

Målet var at kunne lave en klar introduktion, en præcis vejledning og en god modellering, som i alle dele fremstår brugbar og relevant og opfylder succeskriterierne i afsnit 2.2, for dermed at have faciliteret implementationen af vejledningen og brugen af denne positivt i et virksomhedsmiljø. Desuden skulle den skabe en værdiforøgelse for producenten, som efterfølgende ville kunne lave en handlingsplan for hvordan sikkerheden øges i produktet.
Vores metodik kan (præcis som de præsenterede, lignende modeller) bruges som et billigt og effektivt værktøj til at undersøge, hvor problemerne potentielt vil være at finde for design- eller udviklingsafdelingen og give dem idéer til videre handlinger, som kan løse disse. Desuden kan det senere hen bruges som et værktøj ifm. pen-testing processen til at identificere sårbarheder. Jf. ovenstående er det tilfældet og vi har til fulde opfyldt det mål og de underliggende kriterier, som vi opstillede i starten af projektet.

A Vejledningen

Forfatter: Rasmus

Denne vejledning tjener til at formalisere en metode, hvormed en producent af et stykke moderne forbrugerelektronik, kan kortlægge de potentielt sikkerhedsmæssigt problematiske sider af sit produkt.
Målgruppen er defineret ved den mængde af produkter, som ikke tidligere har været integreret med almindelige, trådløse teknologier rettet mod forbrugere, men qua den rivende innovation på området, i dag implementerer disse for at tilbyde nye features. Eksempler på dette kan både være et simpelt "intelligent" køleskab med internetadgang, et avanceret TV eller et klimasystem. Fællesnævneren er, at man typisk ikke tidligere har tænkt den helt store grad af sikkerhed ind i disse produkter og samtidig har produkterne ikke været udsat for angreb i samme grad som computer- og databasesystemer, hvor det almene fokus og mængden af værdifuld information typisk er større.

Metoden er tænkt som et værktøj for en producent af et produkt i ovenstående kategori, der med dette kan få klarhed over hvor sikkerheden halter i produktet og dermed undgå utilsigtet interaktion af produktet (f.eks. afsløring af fortrolig information om produktet selv eller data opbevaret i produktet eller uhensigtsmæssig brug).
Det forudsættes, at vedkommende som udfører denne undersøgelse, har en vis portion viden om produktets funktion (herunder anvendte forbindelsesteknologier, dataveje og viden om vigtigheden af enkelte kompontenter) samt et grundlæggende kendskab til begreber og angrebsmønstre indenfor datasikkerhed.

Den metodiske fremgangsmåde i vejledningen er bygget op over fem trin, som udføres i rækkefølge. Hvert trin består af et centralt spørgsmål, som kendetegner trinnets formål og indhold. Herefter følger en kort beskrivelse af selve trinnet: Hvad der skal kortlægges og hvordan; endvidere gives referencer til værktøjer, såfremt dette er nødvendigt.
Det gøres klart, at processen med undersøgelsen er dynamisk; således kan der undervejs opstå et behov for at vende tilbage til et tidligere trin og revurdere sine resultater. Det er ikke atypisk, men et resultat af det ofte dynamiske (trussels-)miljø, som omgiver os.
Efter hvert trin findes et underafsnit indeholdende en motivation, samt en forklaring om trinnet og hvorfor det er er relevant i kontekst af den hverdag og det miljø, som produkterne i målgruppen befinder sig i.
Dette er tiltænkt til at fungere som en inspiration til arbejdet med det enkelte trin og vil i øvrigt kunne benyttes ved dialog med interessenter med ingen eller meget lille teknisk indsigt.

Vejledningen er suppleret med et forslag til en modellering af de fundne resultater, som skal

facilitere undersøgelserne fremadrettet samt bidrage væsentligt til overskueligheden og anvendeligheden af resultaterne. Der henvises løbende fra metoden til de tilsvarende trin i modelleringen, der i sin helhed findes i afsnit A.3.

A.1 Hvad er sikkerhed?

Forfatter: Daniel

For at forstå og kunne bruge denne vejledning er det vigtigt at vide hvad sikkerhed er.
Inden for datasikkerhedsfaget bruges tre sikkerhedskriterier typisk:

Confidentiality Der skal garanteres total fortrolighed – både hvad angår læsning og skrivning af data, både i opbevaring og transit.

Integrity Det skal garanteres, at data ikke kan modificeres eller slettes af uvedkommende parter.

Availability Selvom systemet skal være sikkert, skal der også være adgang til det, når der er behov for det og systemet skal samtidig fungere optimalt.

Disse tre principper er kendt sammen som *the CIA-properties*. Målet med datasikkerhed er at få opfyldt disse kriterier.
Det er også vigtigt at forstå hvordan et angreb er skruet sammen. Groft set er der to dele af et anbreb. Det handler oftest først og fremmest om at få indhentet information på den part som angribes. Det kan f.eks. foregå ved fysisk besøg, indhentning af oplysninger på hjemmesider og sociale medier og undersøgelse af IP-adresser mv.
Dernæst kan der bruges automatiske eller semiautomatske værktøjer til at søge efter sårbarheder i netværk. Dernæst kan den samlede viden bruges sammen med tidligere erfaringer til at finde sikkerhedshuller og til slut kan disse bruges til et angreb, hvor værdier i form af eksempelvis data kan udtrækkes. Vær dog opmærksom på, at der også findes hackere, hvis incitament blot er at få lagt dit system ned!
Det anbefales, at brugeren af vejledningen sætter sig ind i hvad det er, der skal sikres imod. Hvis ikke dette gøres vil selv den bedste gennemgang af vejledningen være spildt arbejde.
Det er vigtigt at notere sig, at total sikkerhed altid er målet, men aldrig nåes.

Misbrug Begrebet går igen i vejledningen. Det dækker over andet end generel "hacking". Misbrug dækker, som ordet i sig selv siger, al brug af produktet som ikke er den brug produktet er beregnet til. Det betyder, at misbrug af et smart køleskab kan være at ændre temperaturen, misbrug af en smart kaffemaskine kan være at lade være med at varme vandet op, mens misbrug af en smart støvsuger kan være at få den til at puste i stedet for at suge.
Der er altså ikke nødvendigvis tale om større kriminelle handlinger, som adgang til følsomme informationer eller overtagelse af identitet, men ligeså vel kompromittering af systemets

"grænser", som kan medføre skader på produktet eller forbrugeren eller udføre handlinger, som overtræder lovgivning eller lignende.

A.2 Metodik

Forfatter: Fælles

Herunder følger de individuelle trin i metoden.

A.2.1 Hvilke inputs er der på produktet?

Først og fremmest skal de enkelte inputs i produktet identificeres. Det sker for at skabe overblik over forbindelserne mellem forbrugeren og produktet, som på et eller andet sæt kan mediere interaktion imellem disse. Med produktet in mente, nedfældes en liste af samtlige inputs, som er repræsenterede i det givne produkt/system.
For at simplificere identifikationen, anbefales det at opdele i trådløse hhv. kablede forbindelser. Herunder følger en ufuldstændig liste med eksempler på dette.

Eksempler på **trådløse** forbindelser i et produkt, er:

- IEEE 802.11-forbindelser[53] (Wi-Fi™)
- Bluetooth[54]
- Infrarød[55]
- RFID[56] (f.eks. NFC[57])
- GSM/EDGE/UMTS/LTE[58]

Eksempler på **kablede** forbindelser i et produkt, er:

- Servicestik (div. mere eller mindre standardiserede typer; f.eks. CAN[59])
- CEC[60] (kører via HDMI, men tilføjer yderligere funktionalitet)
- USB™[61]

[53] http://en.wikipedia.org/wiki/802.11
[54] http://en.wikipedia.org/wiki/Bluetooth
[55] http://en.wikipedia.org/wiki/Infrared#Communications
[56] http://en.wikipedia.org/wiki/Radio-frequency_identification
[57] http://en.wikipedia.org/wiki/Near_Field_Communication
[58] http://en.wikipedia.org/wiki/List_of_mobile_phone_standards
[59] http://en.wikipedia.org/wiki/CAN_bus
[60] http://en.wikipedia.org/wiki/Consumer_Electronics_Control#CEC
[61] http://en.wikipedia.org/wiki/USB

- IEEE 1394[62] (Firewire™)
- Antennestik[63]/RF connector[64]
- Strøm(-forsyning)
- Video
  - ⋆ HDMI™[65]
  - ⋆ VGA/SVGA/XGA[66] (eller andre analoge computer display standarder)
  - ⋆ DVI™[67]
- Ethernet[68]
- Fysisk interface (f.eks. et alfanumerisk tastatur)
- Optiske forbindelser (f.eks. S/PDIF[69] eller fiber[70])
- Kortlæsere[71] (f.eks. Smart Cards eller memorykort)
- Andre kabler internt mellem komponenter (aflytning/data manipulation) (f.eks. signalpins eller ved protokoller som LonTalk®[72])

Nogle af ovenstående kablede forbindelser er klart outputs fra systemet (VGA, DVI, S/PDIF), men lige præcis disse findes der angreb mod. Derimod er et output som S-video ikke andet end ren video. Forskellen er, at der ved de førstnævnte kan manipuleres med data som flyder tilbage til systemet og derved gøre skade.
Som det allersidste eksempel lægger op til, er det vigtigt ikke kun at kigge på forbindelser som er tiltænkt forbindelser udadtil, men også være opmærksom på forbindelser *mellem* komponenter! Der findes på nogle produkter eksempelvis mulighed for direkte at emulere et datainput eller på anden vis stimulere systemet — en funktion som kan være uhensigtsmæssig for den daglige operation!
Overvej hvorvidt der kunne findes et servicestik bag en aftagelig plade eller en trådløs funktion,

[62] http://en.wikipedia.org/wiki/IEEE_1394
[63] http://en.wikipedia.org/wiki/TV_aerial_plug
[64] http://en.wikipedia.org/wiki/RF_connector
[65] http://en.wikipedia.org/wiki/Hdmi
[66] http://en.wikipedia.org/wiki/Computer_display_standard
[67] http://en.wikipedia.org/wiki/Digital_Visual_Interface
[68] http://en.wikipedia.org/wiki/Ethernet
[69] http://en.wikipedia.org/wiki/S/PDIF
[70] http://en.wikipedia.org/wiki/Optical_fiber_cable
[71] http://en.wikipedia.org/wiki/Card_reader
[72] http://en.wikipedia.org/wiki/LonTalk

som kan slås til i firmwaren?

Tre spørgsmål kan hjælpe yderligere til at identificere forbindelserne:

- Hvilke muligheder er der for forbrugeren for at interagere med produktet?
- Hvilke muligheder er der for producenten for at interagere med produktet før slutsalget?
- Hvilke muligheder er der for producenten for at interagere med produktet efter slutsalget?

Alle svarene til ovenstående spørgsmål er lig et eller flere inputs, som giver disse muligheder. De skal alle med i modellen.

For at facilitere kortlægningen af forbindelserne og som en hjælp i de efterfølgende trin, anbefaler vi at modellere de identificerede inputs vha. Visio og vores model. For dette trin følges vejledning i sektion A.3.1.

Motivation Dette skridt kan virke både indlysende eller irrelevant; "Der er jo kun er en bluetooth-forbindelse i produktet!" eller "Det handler om vores data og ikke dets VGA-stik!", men idéen med denne kortlægning er at kortlægge, *hvad* der gør en service eller data tilgængelig udadtil.
Det er irrelevant at prøve at se på en given delkomponent af produktet, hvis den i øvrigt ikke kan nås udefra eller de redskaber eller den viden der kræves herfor, vil være en langt større tærskel end for andre services. Samtidig kan der potentielt være uendelig mange af disse underliggende komponenter eller services, alt efter hvilket produkt vi kigger på.
Dette trin skal også bruges til at kortlægge de "skjulte" indgange, der som standard ikke er synlige eller tilgængelige for forbrugeren og for hvilke man ikke har indtænkt sikkerheden fra starten. Disse er også i høj grad ønskværdige at finde af netop denne årsag!
Identificeringen tjener også til, i sammenhæng med en vurdering af de hertil forbundne services og deres værdi, at der kan præsenteres en samlet sikkerhedsvurdering af produktet dækkende hele produktet.

A.2.2 Hvilke funktioner/services findes i produktet?

I dette trin identificeres hvilke services eller funktioner der eksisterer i produktet. Med udgangspunkt i modelleringen af trin 1 som inputs, udfyldes "indersiden" af boksen nu med komponenter; se trin 2 af modelleringen (A.3.2).
De services som der her tænkes på, dækker over en relativ arbitrær mængde af indre komponenter, som varierer væsenligt fra produkt til produkt. Med "funktioner" tænkes en gruppering af features, som udføres af den samme delkomponent i produktet eller dækker over features,

som er så godt som identiske i deres funktion.
For at kunne identificere disse, bør man først gøre sig tanker om hvordan produktets styring vs. tilstand er opbygget. Matrixen på figur 16 repræsenterer denne adskillelse.
I et system med central styring og central tilstand, vil en art controller have præcedens over de eventuelt andre komponenter, reagere på input og dirigere output. I et sådant system, vil man typisk kunne "se" (interagere/manipulere) alle (eller langt de fleste) services fra alle inputs.
Dette skal ses i modsætning til et produkt, hvor kontrollen er distribueret; her har de enkelte komponenter mulighed for handling — uafhængigt af hvad en anden del gør. Dette ses typisk i mere komplekse produkter, som f.eks. en bil. Dataen er central, idet der her deles info om bilens hastighed, retning, motordata mv., som flere komponenter kan have brug for i deres funktion, men air condition, soltag og lygter er uafhængige og styres ikke af den samme centrale processor.
I dette system vil man typisk ikke have delt interaktion mellem flere inputs og services.
Eksempler på hvad de enkelte funktioner/services i et produkt kan være, er:

- Kanaltuner (TV/radio)
- Strømstyring
- Apps (3. parts kode)
- GPS
- Internetfunktion (-browser)
- Sensorer
- Temperaturstyring
- Generel mekanisk handling (låsemekanisme, støvsugning, kaffebrygning, isterningproduktion)

Det kan være vanskeligt at identificere de forskelle funktioner helt præcist, men tænk over, hvad der kan indstilles/manipuleres/aflæses fra hvert enkelt input og notér dette. Hvis det ikke allerede er klart på dette tidspunkt, hvordan de er delt op i forskellige services, så saml dem i lignende grupperinger og giv dem en etiket.

Modelleringen af disse funktioner beskrives nærmere i afsnit A.3.2, hvor der ligeledes forefindes eksempler på to produkter i hver sin kategori af state/control-matrixen og deres funktioner/-services.

Motivation Dette trin er med til at skabe en generel gennemskuelighed over hvilke komponenter, som gemmer sig inde i produktet og dermed skabe et grundlag for at kunne differentiere risikovurderingen for de enkelte input, på baggrund af hvad de er forbundet med, senere hen. Identifikationen af services er vigtig, fordi vi efterfølgende kan gennemføre en vurdering af hvor kritisk de enkelte komponenter er for det samlede produkt og så, i sammenhæng med en vurdering af de(t) forbundne input(s), kan skabe et samlet risikobillede. Det er imidlertid vanskeligt at lave en generel procedure for så mange forskellige produkter for lige præcis denne del af metoden, men der må henvises til de givne eksemplers generelle applikationer.

A.2.3 Hvordan er input og services linket sammen?

Dette trin har til formål at kortlægge forbindelserne mellem de enkelte services/funktioner og inputs, således, at det kan anskueliggøres, hvad der kan aflæses fra de enkelte inputs.
Iht. state/control-modellen (figur 16) overvejes relationen mellem inputs og funktioner her jf. trin 2 i modelleringen (afsnit A.3.2). Central kontrol vil eksempelvis være kendetegnet ved, at alle inputs på en eller anden måde kan gå ind og manipulere ved funktionerne, fordi de styres gennem én central enhed. I den forbindelse er det muligt, at der er kodet en begrænsning mellem de forskellige funktioner i controlleren, men det er netop her, at der kan opstå en sikkerhedsrisiko, for *hvis* dette ikke er udført ordentligt, vil man som producent typisk ikke opdage det, fordi man netop opfatter det som i praksis adskilte funktioner.
Den samme proces gør sig gældende for et produkt med distribueret kontrol. Her kan man have den opfattelse, at alle inputs er lige vigtige, mens virkeligheden er en anden og det kun er to inputs som er koblet med de helt centrale funktioner i produktet.

Selve trinnet er her, at se på hvad der kan opnås forbindelse til fra hvert input (hvilket er nemmere end at se fra funktion og så finde alle de inputs der er forbundne til denne), ved eksempelvis at udnytte ens viden om produktets funktionalitet (eller se i brugermanualen) eller direkte at forbinde sig på hvert input.
Vær opmærksom på, at denne metode kan lede til en revurdering af resultaterne af metodens trin 2 (A.2.2), hvis ens forkendskab til produktet er mindre end antaget.

For eksempler på hvordan koblingerne kan implementeres i modellen, henvises til afsnit A.3.3.

Motivation Med introduktionen af skelnen mellem produkttyper ved state/control-matrixen, bruges denne overvejelse til at fastslå forbindelserne mellem den distribuerede eller central styring eller tilstand. Der skabes en eksplicit klarhed for hvordan interaktionen er mellem de enkelte inputs og de indre funktioner/services i det givne produkt.
Modelleringen af forbindelserne er vigtig af ovenstående årsager og selvom de kunne stå som

en del af trin 2, så kan en lidt anderledes metode end ved identificeringen af funktionerne, forplumre billedet og eventuelt bidrage til forvirring. Man vil i de fleste tilfælde kunne slippe relativ let over dette trin.

A.2.4 En samlet risikoanalyse

I dette sidste trin af metoden, foretages en *trussels- og risikovurdering* af de i trin 1-3 beskrevne og modellerede input og funktioner med henblik på at give et endeligt billede af hvilke sikkerhedsrisici, der findes i det pågældende produkt og især hvilke, der bør tages hånd om, for at undgå misbrug!
Risikoanalysen kan opdeles i tre trin, som er angivet herunder:

A.2.4.1 Hvor kritiske er de forskellige funktioner/services? I dette trin skal der laves en vurdering af konsekvensen ved misbrug af de i produktet eksisterende funktioner.
Aspekter som bør overvejes, er f.eks.:

- Er der data i en funktion, som ikke må deles?
  Er brugernes data beskyttet mod misbrug (evt. iht. lovgivningen på området)? Er deres identitet?
  Kan forretningshemmeligheder blotlægges?

- Hvilken indvirken kan det have på produktet og dets funktion, at en given service eller data bliver tilgået eller modificeret? (tænk på tilgængelighed af produktets funktion ift. "Denial-of-Service" og at produktet altid skal være tilgængeligt og funktionsdygtigt)
  Er det et krav fra tredjepart, at produktet skal være tilgængeligt?

- Kan brugeren ændre produktets formål, udover hvad det er godkendt/sikret til eller ændre data og indstillinger, som ikke må ændres? (Eksempelvis en landeindstilling til kontrol af copyright-tilladelser på afspillede medier eller ændre i en brugerdatabase)

- I hvor høj grad står virksomhedens renomé på spil i tilfælde af en kompromitering? (Integritet, tillid mv.)

Tildel, ud fra disse overvejelser, en *impact score* på en skala fra 1 til 5 til hver funktion, hvor 5 er mest kritisk og 1 den mindst kritiske.
Denne score er relativ og gælder kun for det aktuelle produkt. Idet der ønskes at give vejledning til *hvor* de største sikkerhedstrusler i produktet findes, gives der derfor ikke en specifik skala. Skalaen går fra 1 for produktets mindst kritiske funktioner og til 5 for de mest kritiske. Vurderingen er helt op til analyseteamet ud fra de ovenstående spørgsmål.
For hver service kigges på de forbundne services. Hvis en forbunden service har en større score, tildeles den pågældende service samme score, således at forbundne services har den højeste

score i blandt sig. Gå dernæst ud ad kanalerne til de inputs som faciliterer kommunikation med disse funktioner og tildel denne den samme score. Hvis der allerede står en score på dette input, som er højere, ændres denne ikke, men ellers er den nye score den gældende.
Gentag for alle funktioner og deres inputs.

A.2.4.2 Hvad er truslen mod hvert input? Analysér hvert input i forhold til hvor kendt og hvor dokumenteret det er. Desto mere udbredt og dokumenteret, desto større kendskab til mulige angreb. F.eks. ligger der frit tilgængelige guides på internettet, til hvordan man bryder sikkerheden på et almindeligt WiFi-netværk, ligesom bluetooth også kan knækkes vha. lettilgængelige guides. Det er derimod sværere at kompromittere fysiske/kablede forbindelser, da det kræver en fysisk nærhed, som ikke nødvendigvis kan opnås på simpel vis.

Ud fra disse betragtninger tildeles hvert input en score fra 1-3, hvor 1 gives hvis det er en fysisk og mindre kendt forbindelse, 3 gives hvis det er en kendt trådløs forbindelse med en omfattende, tilgængelig dokumentation og 2 gives til alle "normale", fysiske inputs, som kan antages at være en god angrebsvektor samt trådløse inputs, som har mulighed for at sende en meget smal mængde data eller med kun lille indvirken og som der derfor ikke i samme grad kendes angreb gennem.

Det er vigtigt, at der i dette trin ikke skeles for meget til størrelsen af sandsynligheden for angreb på det pågældende input er. Selvom synligheden af inputtet er minimal, vil en grundig hacker stadig kunne finde det og der skal derfor stadig gøres noget for, at der ikke kan læses direkte herfra. Når det kommer til sikkerhed, er det vigtigt at være tilstrækkelig paranoid, når det kommer til hvad uvedkommende vil og kan gøre [8]. Derfor kan det blive kritisk, hvis sikkerheden på et input forkastes, fordi der historisk set ikke er en trussel mod dette. Her kan det være en fordel at kigge på, om der er mulighed for at simplificere de kristiske dele af produktet og sørge for, at mængden af kritiske dele, er så lille som mulig.

A.2.4.3 Er der beskyttelse på disse? Dernæst undersøges den tillagte sikkerhed på hvert input. Det er vigtigt at få blotlagt alle aspekter af sikkerheden og ikke kun om den er krypteret. Analysen af den tillagte sikkerhed skal også kigge på andre dele af sikkerheden. En god krypteringsalgoritme kan nemt knækkes, hvis den nøgle man bruger, ikke er ordenligt genereret. Hvis man bruger en random number generator, er det vigtigt at benytte en, som er certificeret til brug til kryptering og ikke bare den pakkeløsning som følger med produktet, da en standard, eller decideret dårlig number generator, relativ nemt kan knækkes.
Nøgleudveksling er også en stor del af sikkerhedsaspektet. Bruges der en kendt, sikker protokol (Diffie-Hellman f.eks.) eller bruges der en mere usikker løsning, der kan brydes uden større matematisk kunnen? Hvis det sidste er tilfældet, kan hverken stærke nøgler eller krypteringsalgoritmer hjælpe. *Et sikret system er kun så stærkt som det svageste led* [8]. Dette må aldrig

glemmes, da det kan have store konsekvenser. Hvorfor skal en angriber forsøge at knække en stærk kryptering, hvis brugerne af systemet bruger svage kodeord?
Selvom producenten af det pågældende produkt ikke selv er ekspert dette område, er det vigtigt, at få det undersøgt grundigt. Det kan derfor være en fordel, at få en kapacitet med en større viden på området, til at hjælpe med at analysere sikkerheden på hvert input.

Hvert input tildeles en score på 1-3, hvor 1 er en ingen sikkerhed.
Scoren 2 gives hvis der er tænkt over sikkerheden, men at den måske er baseret på en ældre krypteringsalgoritme eller andre former for sikkerhed, som forholdsvist let kan brydes, men dog stadig vil være en udfordring på vejen til at bryde kompromitere inputtet.
Scoren 3 gives hvis der foreligger et robust sikkerhedsystem på det pågældende input, her skal producenten formentlig have haft kontakt til en ekspert på området ved udviklingen af produktet.
Denne score divideres med produktet af de to foregående, således at den samlede score bliver mindre, desto større sikkerhed der inkorporeret på det pågældende input.

Når alle disse trin er gennemarbejdet og analyseret, er det vigtigt at få lavet en fremstilling, der nemt giver et godt overblik. En lang liste på punktform kan være svær at overskue, så det anbefales at modellere produktet som en kasse og bruge farvekoder til hvert input og hver service og funktion. På denne måde bliver det hele overskueligt og en tekniker der får præsenteret modellen kan med det samme se, hvor udfordringerne ligger (den røde farve antageligvis); se eksempelmodelleringen med farver i figur 22.

Motivation Her benytter vi os af gængse it-sikkerhedstekniske principper, kaldet *the CIA properties*[73], hvor man undersøger konsekvensen af en given komponents kompromittering, frekvensen af angreb imod denne og endeligt de foranstaltninger som er gjort, for at forhindre disse angreb.
Samlet set kan vi så danne os et billede af hvor på produktet, de største risici er tilstede og efterfølgende tage aktion på dette.

A.2.5 Modtræk

I dette afsnit præsenteres en række forslag til hvordan man i det videre arbejde, kan tage hånd om de fundne risici i modellens forrige punkter .
Modellens formål er, som beskrevet, at afsløre de mest sandsynlige usikre dele af produktet/-systemet, men disse sårbarheder kan manifestere sig vidt forskelligt afhængigt af det konkrete produkt.

[73] http://www.techrepublic.com/blog/it-security/the-cia-triad/

Herunder præsenterer vi således, ud fra almentkendte sikkerhedskriterier (som givetvis vil vække genkendelse), små eksempler på, hvordan de enkelte typer sårbarheder kan manifestere sig i systemet og hvad man eksempelvis kan gøre for at afhjælpe disse.
Kriterne dækker en bred kam af trusler, men listen er uudtømmelig og derfor holdt på et generelt niveau ift. hvordan de konkret forekommer i systemet — både i inputs og funktioner!

Det anbefales at læse forslagene i afsnit A.2.5.1 grundigt igennem og lade sig inspirere af disse. Undersøg dernæst om de mest kritiske dele af det pågældende system, er påvirket af disse trusler og hvordan de kan sikres.
Det anbefales endvidere at bevæge sig systematisk frem og starte med de mest udsatte inputs og services først. Man kan evt. tage hensyn til hvilke grundighed, man vil behandle de enkelte sårbarheder med. Afhængigt af mængden kan man derefter arbejde sig nedad i klassificeringerne under hensyntagen til den generelle risikovillighed hos producent og mængde af inputs.
En eventuel prioritering af hvilke inputs man vil taget fat i først, bør ske med skelen til produktets funktion, således, at man som minimum sikrer, at kritiske dele af infrastrukturen er tilstrækkelig beskyttet og at produktet kan opfylde sin funktion. Det tilrådes dog at starte med de inputs, som har den værste rating i modelleringen.
Den nøjagtige fremgangsmåde hvad angår prioriteringen her, hører sig dog til i et forretningsområdet og bør afvejes ift. firmaets strategi.
Efter processen med opsætning af foranstaltninger, bør modelleringen udføres på ny. Sikkerhedsscoren vil formodentlig have ændret sig, men nogle inputs og funktioner vil stadig fremstå i de mere kritiske kategorier. Det viser dog i virkeligheden blot, at opmærksomheden konstant bør være rettet mod disse, fordi det er her, at de største trusler sandsynligvis vil kunne findes. Imidlertid er der nu reelt foretaget en indsats for at modarbejde disse.

De konkrete mål for denne del af processen, bør være at minimere konsekvensen (dvs. at minimere skaden) ved udnyttelse af de mulige sårbarheder, som er identificeret tidligere.

A.2.5.1 Konkrete foranstaltninger Herunder følger en liste af konkrete sikkerhedstrusler og forslag til foranstaltninger, som kan bruges som referenceliste, når der skal implementeres løsninger på de identificerede trusler.
De er kategoriseret efter tre segmenter: Hardware, software og data og under hver af dem, oplister vi, på en systematisk måde, forskellige typer af foranstaltninger, som kan minimere risici og konsekvensen af mulige angreb. Nogle af foranstaltningerne kan række ud over produktet og beskæftige sig med dets nære omgivelser.
Foranstaltningerne retter sig ikke kun mod inputs, men også mod services, hvor man imidlertid ikke ville kunne aflæse samme effekt på inputtet ved en reevaluering af metoden.

Der gøres opmærksom på, at listen af foranstaltninger ikke er fuldstændig, men de oplistede trusler dækker som regel det meste og der opfordres til, udfra disse, at søge yderligere viden fra relevante kilder: [1, 8, 9, 23, 25].
De fleste løsninger retter sig generelt mod at begrænse adgangen og forstærke den adgangskontrol, som, nogle steder, allerede er i brug. Desuden fordres der til eftersyn af forældede algoritmer i ældre hardware og softwarepakker.

Hardware Truslerne mod hardwaren hænger i mange tilfælde uløseligt sammen med softwaremæssige problemer, i kraft af den kode som afvikles på hardwaren. Herunder gives dog, så vidt muligt, eksempler på løsninger, som kun relaterer sig til hardwaren. Mange af de foreslåede løsninger gælder for flere af trusselstyperne, men præsenteres kun én gang.

- Interruption/Denial of Service
  Interruption forekommer, hvis hardwaren ikke er tilgængelig for brug, når den burde være.
  - ⋆ Installér en UPS, som kan sikre mod utilsigtet strømafbrydelse – bevidst eller ubevidst.
  - ⋆ Hav en redundant internetforbindelse eller mulighed for nemt at genstarte, så man ikke er afhængig af en udbyder. Brug gerne fiber på den ene og kobber på den anden, så det ikke bare løber sammen i skabet ude på vejen
  - ⋆ Sikr at strømforbindelsen er beskyttet – især hvis indstillinger afhænger heraf. Brug et backupbatteri
  - ⋆ Hav redundante servere, så trafik kan redelegeres ved (D)DoS-angreb
  - ⋆ Brug om muligt lukkede systemer, så produktionsservere ikke er forbundet til internettet
- Interception/Theft/Information disclosure
  Interception er tyveri af den fysiske hardware eller tilgang til denne.
  - ⋆ Sørg for, at produktet ikke kan fjernes, hvis det skal placeres stationært
  - ⋆ Begræns adgang til produktets indre med specialskruer el.lign. foranstaltninger
  - ⋆ Find en afvejning af mulighed for udskiftninger fra brugerens side versus total fastlåste elementer (f.eks. lim, specialskruer, "warranty void if removed"-klistermærker o.lign.)
  - ⋆ Brug adgangskontrol til de fysiske placeringer og hardwaren og hav overblik over adgang

- Modification/Tampering
  Modification er ændringer af den fysiske hardware; disse må kun ske af folk med korrekte rettigheder og på korrekt måde. Flere af punkterne fra *Interception* er også relevante her.
  - ⋆ Sørg for at have adgangskontrol til fysiske placeringer ad hardwaren og sørg for at logge hvem der bruger adgangen.
  - ⋆ Sørg for at ændringer af hardwaren kan opdages, f.eks. ved nedlukning af hardwaren, hvis den bliver åbnet.
- Fabrication/Substitution/Spoofing
  Fabrication er udskiftning eller efterligning af hardwaren eller manglende unik identifikation. Ligner på flere punkter de to ovenstående.
  - ⋆ Serienumre eller anden unik ID fra producentens side (f.eks. hologram-/vandmærket mærkat)
  - ⋆ Sørg for løbende at kontrollere at alt er som det skal være i serverrummet, således at kopiering eller udskiftet hardware blivet opdaget.
- Elevation of privilege
  Elevation of privilege (altså manglende authorization) er typisk softwareafhængigt, men kan muligvis forekomme som en ren hardwareimplementation. Vi har ingen eksempler herpå, men henviser til samme punkt under *Software*.
  - ⋆ Hav styr på den fysiske adgangskontrol til hardwaren (brug f.eks. adgangskort og stil spørgsmålstegn ved fremmede "gæster", som vil ind i serverrummet).
  - ⋆ Social Engineering er ofte den store trussel her, hvor udefrakommende snyder sig til adgang eller oplysninger. Tænk over ikke at udlevere passwords eller identiteter til ukendte personer. Hvordan kan du eksempelvis være sikker på, at manden som har ringet dig op, rent faktisk kommer fra din leverandør?

Software Softwaren er ofte den mest udsatte del af systemet, fordi det er nemmere tilgængeligt og har en større udbredelse, sådan at sårbarhederne også er mere eksponerede. På samme måde som ved hardwaren, er sårbarhederne og deres løsninger ofte forbundne; f.eks. vil en log, som ikke kan tilgås pga. et DoS-angreb, ikke være meget bevendt.

- Interruption/Denial of Service
  Interruption forekommer, hvis softwaren ikke er tilgængelig for brug, når den burde være.
  - ⋆ Benyt metoder i netværksudstyr eller hos ISP'en, som kan redelegere (D)DoS-angreb
  - ⋆ Benyt principper fra robust programmering [3] i udviklingsfasen

- Interception/Information disclosure
  Interception af software er ikke et væsentligt problem, men har man ophavsrettighedsbeskyttet kode, kan man være interesseret i skjule den. Oftest er det dog i egen interesse, at kode som skal være sikker også kan kontrolleres af flere personer. White hat-hackere[74] findes der heldigvis også!

  ⋆ Benyt obfuskering på kritisk kode (men vær opmærksom på, at det kan sænke performance væsentligt og stadig kan brydes)

- Modification/Tampering
  Modification er ikke-tilladte ændringer i koden.

  ⋆ Brug hashfunktioner eller lignende metoder til at sikre integriteten af koden

  ⋆ Sanér dine inputs (både fra brugeren og andre dele af softwaren)

  ⋆ Benyt robust programmering (se ovenfor)

  ⋆ Beskyt produktionsservere (servere hvor koden udvikles og afvikles)

- Fabrication/Spoofing
  Fabrication er, hvis koden ikke er hvad den giver sig ud for at være, ved f.eks. at efterligne produktionskode, men er ondsindet. Man skal således kunne identificere "korrekt" kode.

  ⋆ Certificér din kode - både overfor produktet selv og brugeren

  ⋆ Identificér brugere og systemer på en sikker, entydig og korrekt måde

- Elevation of privilege
  Manglende adgangskontrol i softwaren, som utilsigtet kan tillade ændrede rettigheder til en enhed.

  ⋆ Brug en velkendt og sikker implementation af adgangskontrol og hold den up-to-date

  ⋆ Sørg for at have faste rutiner for tilbagekaldelse af rettigheder og gennemgang af nuværende. Brug en overskuelig struktur for filsystemerne, så det bliver nemmere at gøre

  ⋆ Sørg for at brugeren kun har adgang til de dele af softwaren, som der skal bruges

- Repudiation
  Repudiation er hvis en aktivitet kan udføres uden at den kan spores bagefter.

  ⋆ Brug loggingværktøjer og beskyt logfilerne

  ⋆ Brug hashfunktioner eller lignende til at kontrollere filer og data

[74] Modsætningen til en black-hat hacker. Mere på linje med en pen-tester, men ikke nødvendigvis ansat i et firma

Data Data er bl.a. de databaser og den viden, som er skjult bag softwaren og hardwaren. Ligesom med de to forrige, er data i et afhængighedsforhold, som gør, at sårbarheder her, kan udsætte de andre elementer i systemet og omvendt.

- Interruption/Denial of Service
  Interruption eller tab af data kan være et stort problem for den ramte virksomhed

  ⋆ Sørg for at undgå at uvedkommende har adgang til de pågældende data. Dette kan gøres ved at sikre interne netværk. Korrekt opsatte *firewalls* er et godt værktøj mod uvedkommende og derudover er det vigtigt, at trådløse netværk er robuste og har en god beskyttelse.

  ⋆ Hav en fornuftig *password politik* i virksomheden. Et krav om en vis længde på password samt et krav om en hvis standard (f.eks. brug af både store og små bogstaver, tal og symboler) kan være en god idé. Dette skal dog ske med måde. Hvis den almene brugers password skal være meget avanceret ender det på en post-it på skærmen og det er ikke en optimal løsning. Der kan også være krav om at skifte password jævnligt (f.eks. hver tredje måned), men det er igen vigtigt, at det ikke bliver mere avanceret end at brugeren ikke behøver at skrive deres passwords ned.

- Interception/Information disclosure
  Interception, altså uvedkommende adgang til at læse data, kan bl.a. betyde at forretningshemmeligheder bliver kendt. De første trin til bekæmpelsen af dette, er de samme som beskrevet ovenover.

  ⋆ Sørg for at brugere kun har adgang til de data som er relevante for brugeren og kun i den periode det er relevant. Mange steder bliver der ikke holdt overblik over hvem der har adgang til hvad og en bruger som midlertidigt skulle have adgang til en vis mængde data i en lille periode, har måske stadig adgang et år efter

  ⋆ Kryptér eller hash (husk salt) udvalgte databaser

  ⋆ Opbevar ikke kritisk data (hverken din eller brugerens) i klartekst

  ⋆ Kontrollér ældre hardware og softwarepakker for uddaterede algoritmer og deslignende

- Modification/Tampering
  Modification dækker over uautoriseret ændring af data. De første trin til bekæmpelsen af dette er de samme som beskrevet ovenover. Især kan det være en hjælp at holde styr på hvem der har adgang til hvad og sørge for at kun relevante brugere har adgang til relevant data.

  - ⋆ Sørg for at logge al aktivitet på de servere hvor dataen ligger. Dette vil ikke nødvendigvis forhindre modifikation af dataen, men det kan hjælpe til at sørge for at ikke sker igen ad samme vej
  - ⋆ Sørg for at kontrollere alle ændringer f.eks. ved hjælp af hash-funktioner og tjeksummer. På den måde vil en uautoriseret ændring blivet opdaget og det vil være muligt at rulle tilbage til en foregående version

- Fabrication/Spoofing
  Fabrication dækker over utilsigtet tilføjelse eller substitution af data. Dette kan undgås på samme måde som modification, hvis tjeksummerne og hashfunktionerne også sørger for at dække større dele af systemet end bare de enkelte filer.
  - ⋆ Sørg for at passwordpolitik, firewalls og netværksbeskyttelse er up-to-date

- Repudiation
  Repudiation er hvis en ændring af data kan udføres uden at den kan ses efterfølgende.
  - ⋆ Som ved software: Brug loggingværktøjer og beskyt filerne
  - ⋆ Sørg for at bruge hashfunktioner til at sikre data og filer mod uauftoriseret ændring

- Elevation of privilege
  Manglende adgangskontrol ved tilgang til data, som utilsigtet kan tillade ændrede rettigheder til en enhed. Er altid bundet op på en konkret softwareimplementering, så konsultér dette punkt.

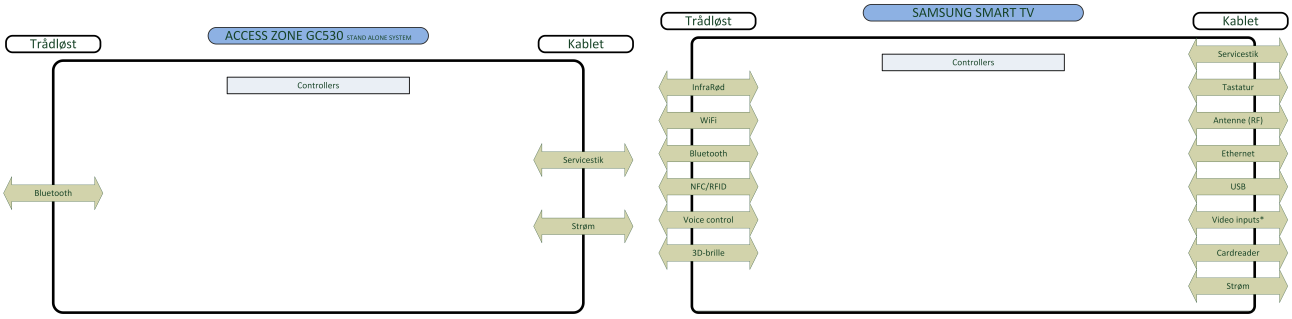
A.3 Modellering

Forfatter: Fælles

Dette afsnit indeholder et værktøj til at modellere et relevant systems input og indre services i relation til de ovenstående trin, for at gøre metoden enklere og mere overskuelig. Der arbejdes, for klarhedens skyld, med to gennemgående eksempler i form af dels et adgangssystem med bluetooth fra MVC-Data ("AccessZone GC530"[75]) og dels et Samsung Smart TV[76].
Til udarbejdelsen er Microsofts diagram- og vektorgrafikprogram Visio benyttet.

A.3.1 Trin 1

Det første trin relaterer sig til vejledningens trin 1 (A.2.1), som bør læses først.
I trinnet skabes overblik over de kablede hhv. trådløse inputs i det givne produkt, men for at gøre det mere overskueligt, foreslås det at modellere det som en "kasse", hvori de fundne inputs forbinder yder- og indersiden.
I figur 15 ses to eksempler på dette i form af de førnævnte systemer fra MVC-Data (figur 15a) hhv. Samsung (figur 15b). Som det ses, er det direkte de i A.2.1 fundne inputs, som er sat ind i



(a) Input i en AccessZone GC530 **(b)** Input i et Samsung Smart TV

Figur 15: Trin 1: Modeller med inputs

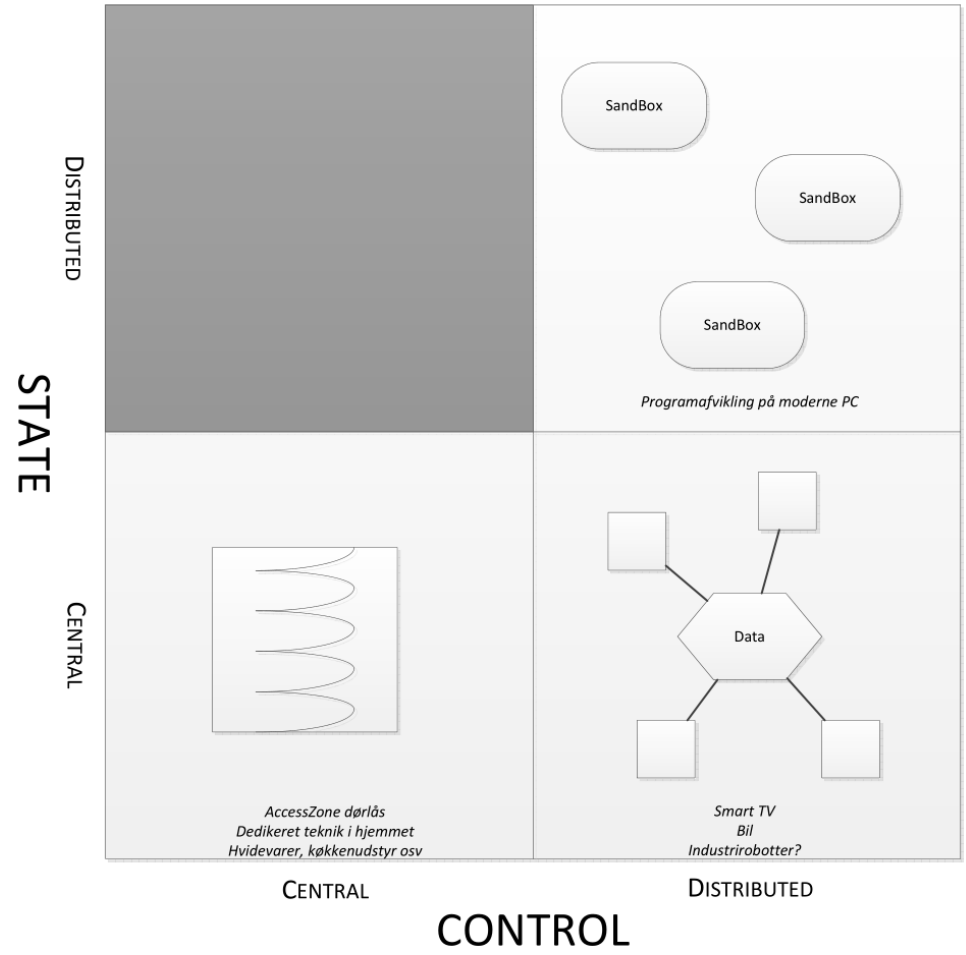
modellen på grænsen mellem "ydersiden" og "indersiden" af den boks som markerer produktets perimeter.
Allerede på denne model ses det at modellen af TV'et kommer til at blive væsentligt mere avanceret end AccessZone modellen. Så meget desto vigtigere er det at der bliver lavet en grundig model, således at der ikke går nogle dele i glemmebogen. Desto flere features et givent produkt har, desto flere ting er der, der kan gå galt.
De mange input i Smart TV'er er af mangt en forskellig karakter. Der er både input der kontrollerer styring af forskellige dele, samtidig er der også input som strøm og billede. På trods af forskelligheden, er det vigtigt at få det hele med for at få karakteriseret alle aspekter af

[75] http://www.mvc-data.com/AccessZone.html
[76] http://downloadcenter.samsung.com/content/UM/201303/20130316093517079/[DAN]X12DVBEUF-0313.pdf

sikkerheden. I eksemplet er alle de forskellige videoinputs (SCART, jackstick, composite video og så videre) samlet under ét input (se figur 15b).

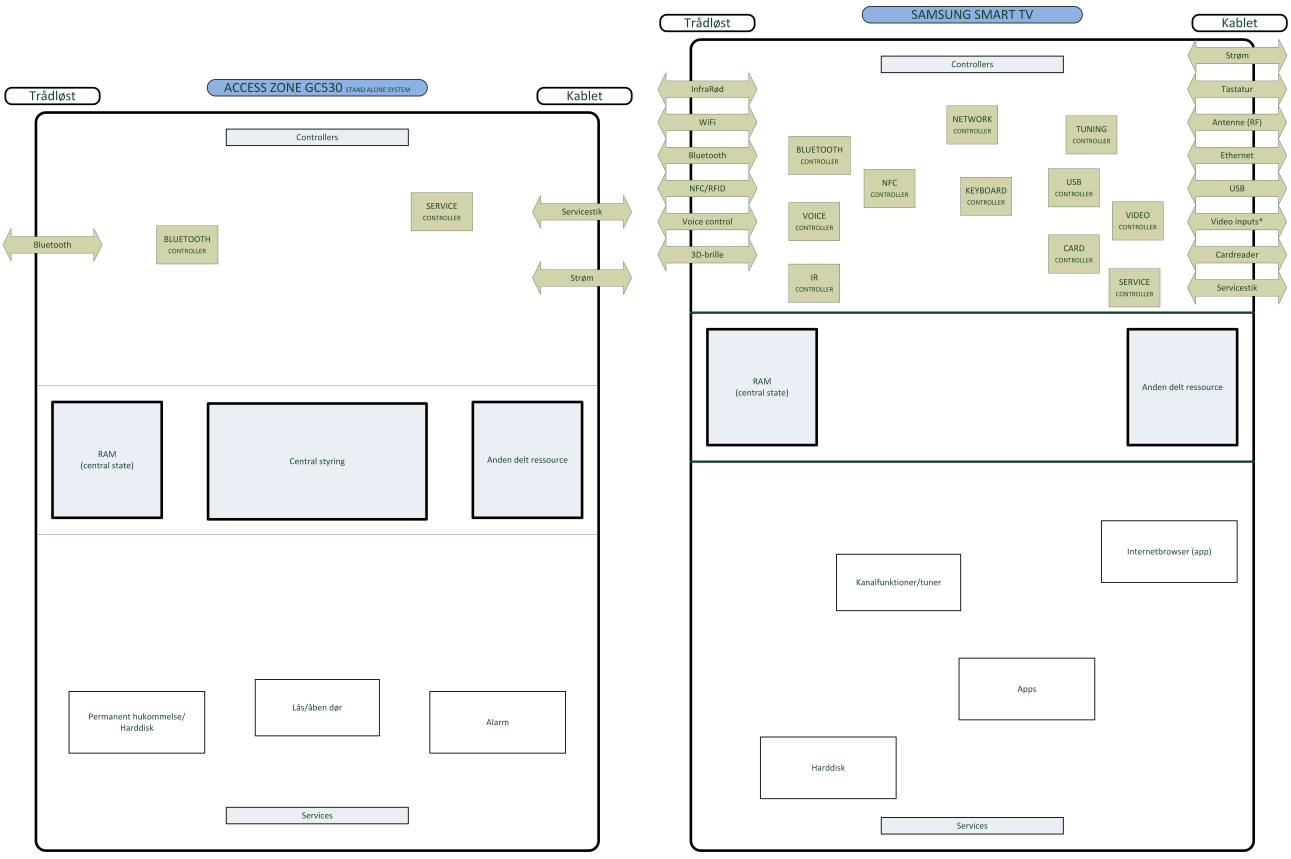
A.3.2 Trin 2

I dette trin indsættes en repræsentation af de services/data, som er identificeret i A.2.2.
Produkttyperne kan i den forbindelse groft opdeles efter tre typer repræsenteret ved en matrix, som det ses på figur 16. Fordelen ved denne opdeling er, at der skabes klarhed over hvordan



Figur 16: Illustration og eksempler på produkters centralt/distribuerede styring og states

forbindelserne mellem inputs og services interagerer, som anskueliggjort i A.2.2.
Det er derfor vigtigt at forholde sig til hvilken af disse tre typer ens produkt bedst relaterer sig til. I de følgende eksempler, bliver det tydelig gjort hvor stor forskellen er på styringen af de forskellige typer af produkter.
På figur 17a og 17b ses eksemplerne med de to gennemgående produkter for modelleringen. Bemærk, at der på modellen af AccessZone-låsen i figur 17a er tilføjet en centralstyring. Baggrunden herfor er, at det vurderes, at både styring/control og state er centralt kontrolleret,



(a) Model af en AccessZone GC530 – central kontrol/centralt state

(b) Model af et Samsung Smart TV – distribueret kontrol/centralt state

Figur 17: Trin 2: Modeller med services og data

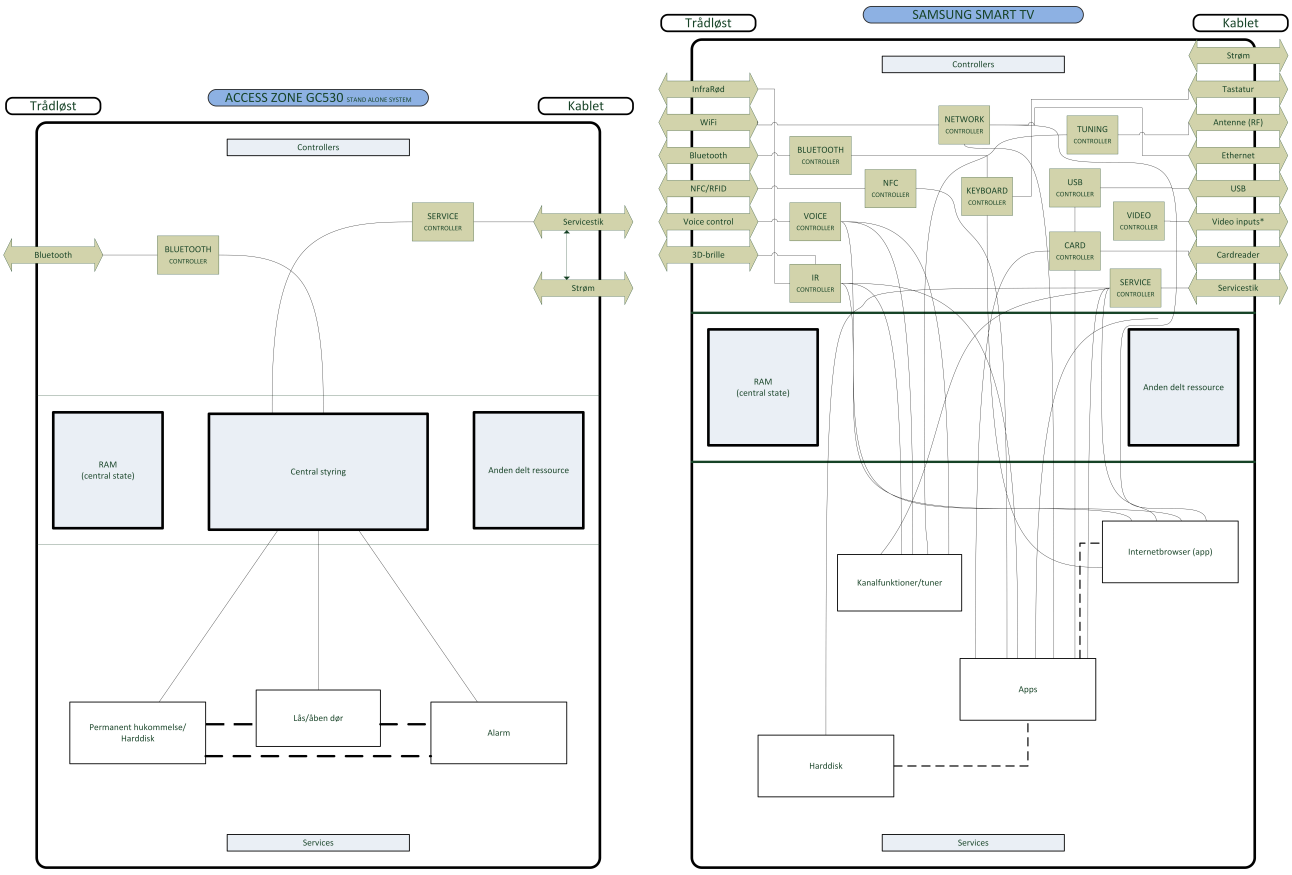
uagtet hvilket input der benyttes. Den samme form for styring/state kontrol ses på andre produkter der har én rimelig fastlåst funktion med få inputmuligheder, f.eks. hvidevarer, kaffemaskiner osv.

Modsat denne styring, findes styringen i fjernsynet der er distribueret, altså hvor de forskellige controllers har direkte adgang til de forskellige services i TV'et, jf. figur 17b. Eksempelvis behøver controlleren til netværket i TV'et, ikke at gennemgå en central styring, der også håndterer kanalvælgeren og vise versa. Samme form ses f.eks. i biler, hvor airconditionanlægget ikke behøver at bekymre sig om hvad bremserne laver.

Den sidste type, som vises i matricen i figur 16, er distribueret state med distribueret control. Et eksempel på dette kan være programafviklingen på en moderne computer (altså ikke selve computeren, men afviklingen). Hvert program har sin egen styring og er i sit egen tilstand, uafhængigt af andre programmer. Et eksempel på en modellering af et sådant system ses i figur 23.

A.3.3 Trin 3

I dette trin analyseres og indsættes forbindelserne mellem de i trin 2 fundne komponenter. Modellen udvides ved, at der tegnes linier mellem inputs, komponenter og services i produktet. Eksempel på dette ses i figur 18. Det anbefales at gå metodisk igennem hvert input/controller



(a) Model af en AccessZone GC530 – Central styring

(b) Model af et Samsung Smart TV – Distribueret styring

Figur 18: Trin 3: Kobling af funktioner/services og inputs i modellen

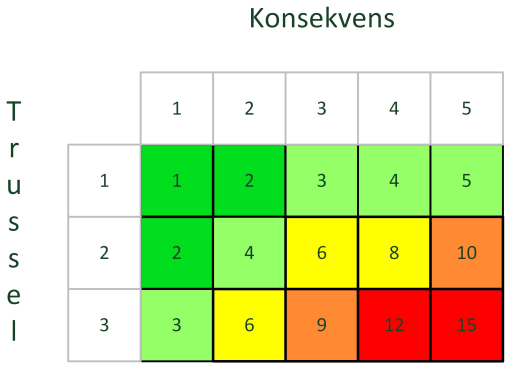
og holde det op imod hver intern funktion.

Bemærk at det i trin 3 er det også vigtigt at få tegnet de indbyrdes forhold mellem de forskellige services og ikke kun mellem input og services! Grunden hertil er, at det er interessant at se, om der er input som har direkte adgang til en hukommelsesenhed, eller om al adgang hertil er via en app eller en anden service.

Der kan opstå tvivl om hvorvidt en given forbindelse eksisterer "fysisk", men kig på den med et kritisk syn og vær sikker på, at der ikke kun er tale om en softwarebegrænsning! I så fald skal forbindelsen stadig på, da en sådan forbindelse potentielt kan udsættes for angreb og aktiveres. Hvis forbindelsen eksisterer mellem f.eks. en databank og et input som et servicestik og et WiFi også udfører lignende servicefunktioner, men umiddelbart ikke har præcis denne adgang, taler det for, at den eksisterer alligevel

A.3.4 Trin 4

I sidste trin af modellen udarbejdes en grundig sikkerhedsevaluering af hvert funktion. Som beskrevet i afsnit A.2.4, startes i "bunden" af modellen og der arbejdes opefter. Alle parametre skal tages i betragtning, både fra hvilke data der ligger i de forskellige services til hvilken form for sikkerhed (f.eks. kryptering) der findes.
For at tydeliggøre resultatet, anbefales det at farvelægge iht. de tildelte scores. I trin 4.1 farves funktioner og controllers med en direkte 1:1 korrespondence mellem de i figur 19 brugte farver, mens gradueringen af inputs i trin 4.2 og 4.3 (hvor skalaen går fra 1 til 15), farves iht. figur 19.



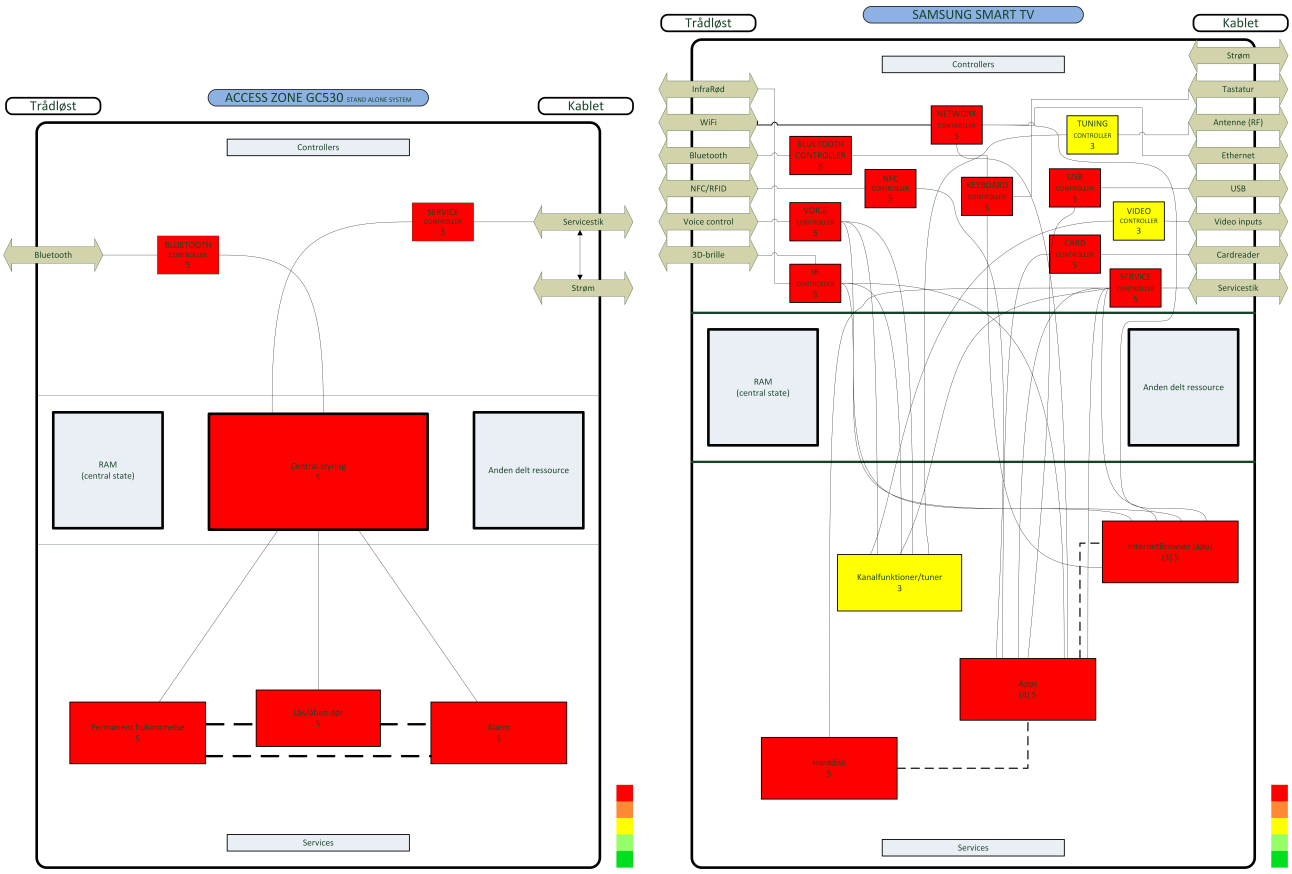
Figur 19: Farvningen af et vurderingen for trin 4.2 og 4.3

Trinnet kan med fordel opdeles i tre deltrin som i metoden:

Trin 4.1 Der startes med at laves en sikkerhedsevaluering af de interne services og funktioner i produktet. Som angivet i afsnit A.2.4 startes der i bunden, arbejdes sidelæns og dernæst opefter, således at en service med en lavere score, får den samme værdi som den højere scorende som service den er forbundet til og at en controller får samme værdi som den mest kristiske funktion, den er forbundet til. To eksempler er givet i figur 20. De brugte farver svarer til dem i figur 19.

Trin 4.2 Dernæst udarbejdes en vurdering af hvert input iht. trin 4.2 (afsnit A.2.4.2). Her antages eksempelvis, at der er en større mængde af brugere, som kender til WiFi end som kender til funktionen af et proprietært servicestik, ligesom det må formodes, at flere kender til brugen af bluetooth end af NFC. Som beskrevet i afsnit A.2.4.2, multiplicerer vi scoren for vurderingen af truslen pr. input (1-3) med værdien for dens controller (1-5) og får derved et tal mellem 1 og 15. Inputtet farvelægges nu efter farveskalaen på figur 19.
Eksempel herpå ses i figur 21.

Trin 4.3 I trin 4.3 ses på den implenterede sikkerhed på inputsne. En score herfor (pr. input; på en skala fra 1 til 3) divideres med den score, som inputtet i forvejen har, som beskrevet

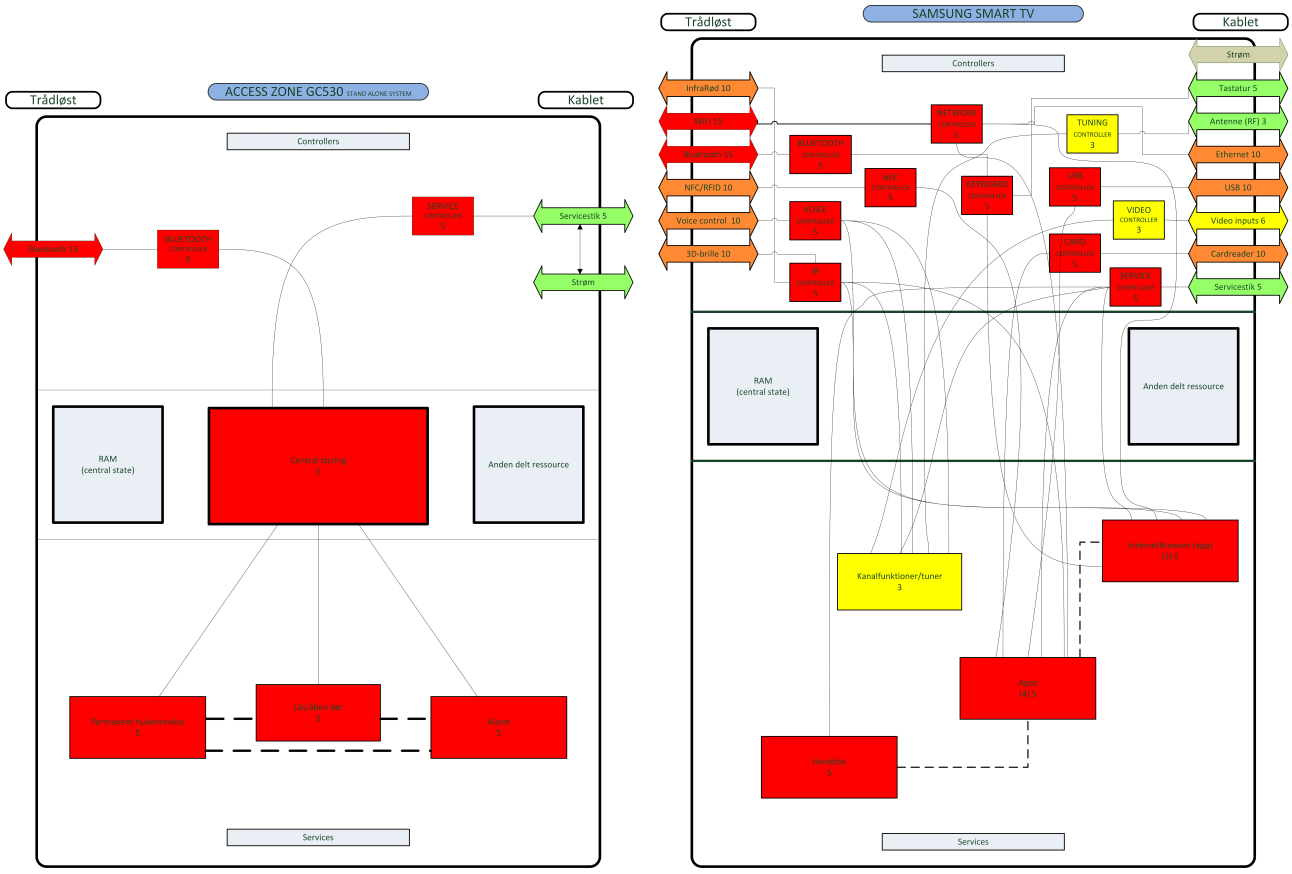


(a) Model af en AccessZone GC530 **(b)** Model af et Samsung Smart TV

Figur 20: Trin 4.1: Angivelse af sikkerhedsrisiko i services og data

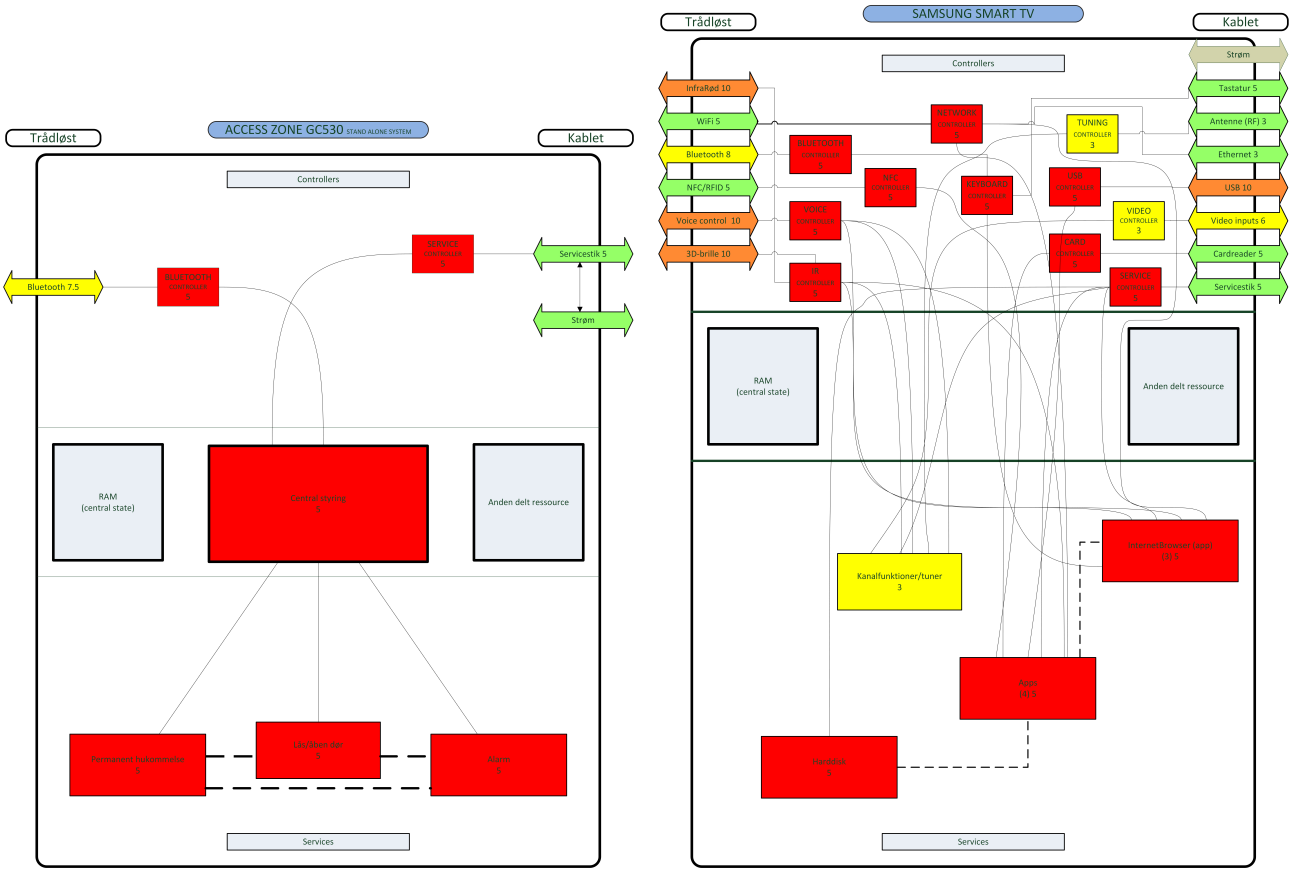
i afsnit A.2.4.3. Inputsne omfarves herefter, så de passer til dette. Farveskalaen følger stadig figur 19.
Hvis udviklingen af produktet i forvejen har haft sikkerhed for øje, bør dette betyde, at de mest kritiske farver på figuren, nedjusteres. Eksemplerne på disse ses på figurerne 22.

Model med distribueret kontrol og distribueret state Som beskrevet i afsnit A.2.2 er der overordnet set tre forskellige grupperinger af produkter, når de grupperes efter state/control, jf. figur 16. Vejledningens eksempler har været henholdsvis et produkt med centralt control og centralt state (dørlåsen) og distribueret control og central state (Smart TV'et). Den sidste kategori dækker over produkter, som har distribueret control og distribueret state. Dette er den mindste gruppe.
Et eksempel er, som beskrevet i afsnit A.3.2, programafvikling på en moderne computer.
Figur 23 viser en abstrakt model af denne form for state/control.



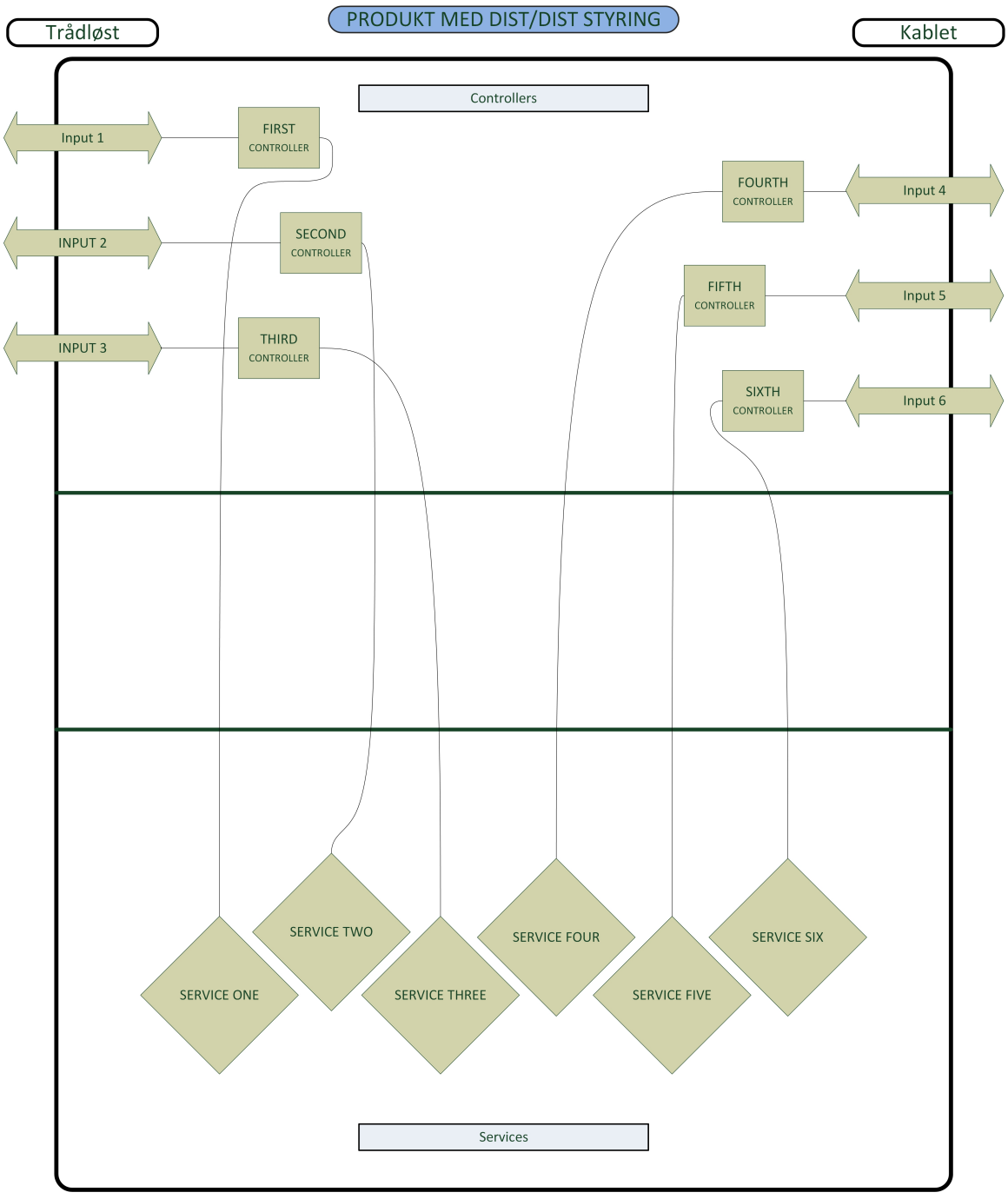
(a) Model af en AccessZone GC530 **(b)** Model af et Samsung Smart TV

Figur 21: Trin 4.2: Angivelse af sikkerhedsrisiko i inputs



(a) Model af en AccessZone GC530 **(b)** Model af et Samsung Smart TV

Figur 22: Trin 4.3: Angivelse af sikkerhedsrisiko i produktet



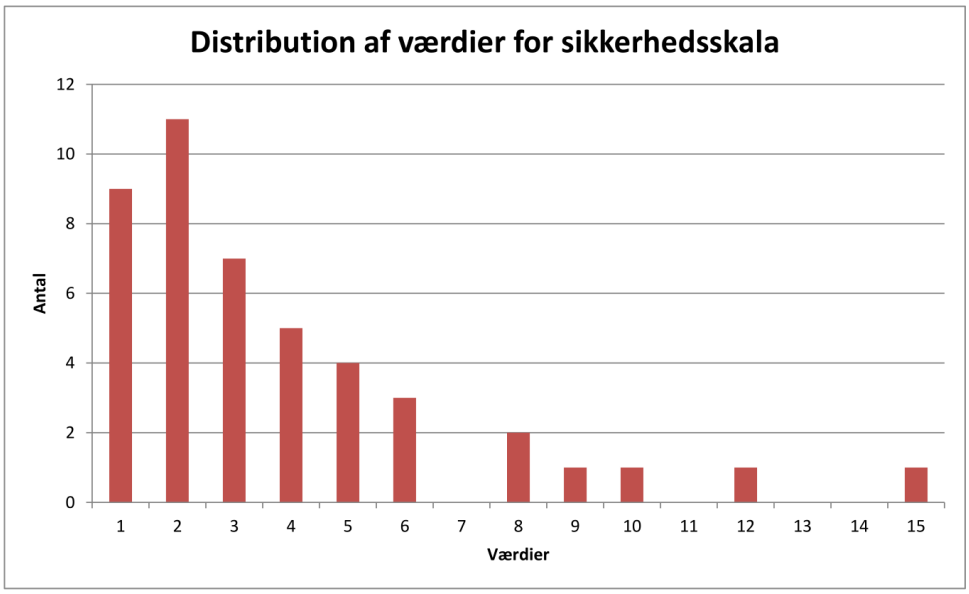
Figur 23: Abstrakt model af et produkt med distribueret state og kontrol

B Appendix

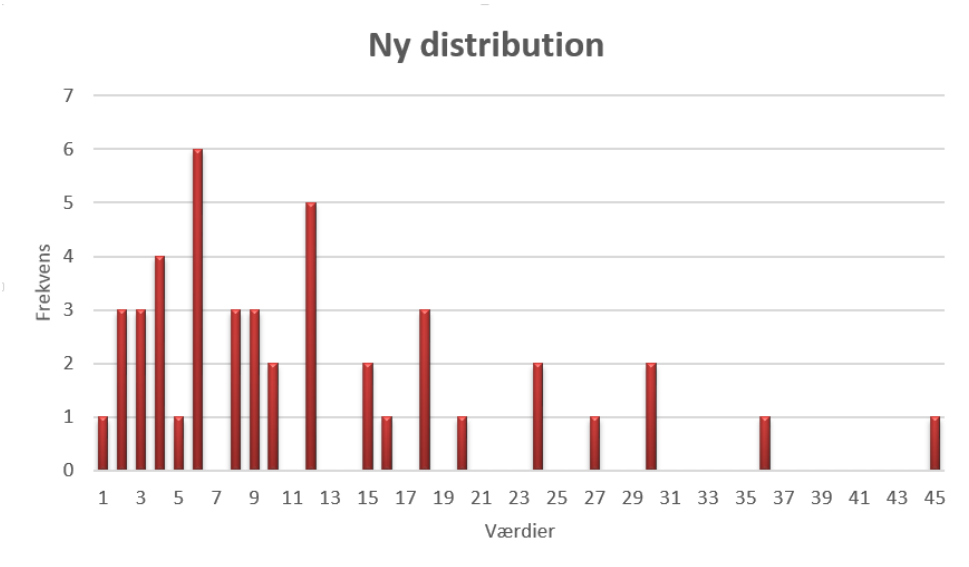
Udregning af farveskalaen for trusselsmatrixen

1	0,333333
2	0,666667
3	1
4	1,333333
5	1,666667
6	2
7	2,333333
8	2,666667
9	3
10	3,333333
11	3,666667
12	4
13	4,333333
14	4,666667
15	5

Figur 24: Figuren viser den udregning, som er brugt til at definere farveskalaen for farvelægningen af funktioner, controllers og inputs i vores modellering. Farverne er rykket et enkelt trin opad for at understrege alvoren ved højtscorende elementer.



Figur 25: Figuren viser distributionen af alle de mulige værdier efter første at have multipliceret et tal på en skala fra 1 til 5 med et tal på en skala fra 1 til 3 og dernæst divideret dem med et tal på en skala fra 1 til 3. Den svære overvægt mod de lave værdier er tydelig.
Skala: 1 – 15. Gennemsnit: 3,82 (ca. 25,5 % af maks). Median: 3 (20 % af maks.)



Figur 26: Figuren viser distributionen af alle de mulige værdier efter samme skala som figur 25, men hvor man i stedet for at dividere med den sidste score, multiplicerer.
Skala: 1 – 45. Gennemsnit: 12,27 (ca. 27,3 % af maks.). Median: 9 (20 % af maks.)

Litteraturliste

[1] Social engineering. URL `http://en.wikipedia.org/wiki/Social_engineering_(security)`. Online; tilgået 30/6 – 2014.

[2] Unified modeling language. URL `http://en.wikipedia.org/wiki/Unified_Modeling_Language`. Online; tilgået 29/6 – 2014.

[3] Matt Bishop. Robust programming. Lecture notes fra ECS 153, Department of Computer Science, University of California, January 2003.

[4] *Vejledning i projektledelse — ISO 21500*. Dansk Standard, first edition, September 2012.

[5] David LeBlanc. DREADful, August 2007. URL `http://blogs.msdn.com/b/david_leblanc/archive/2007/08/13/dreadful.aspx`. Blogindlæg på MSDN; tilgået 2/7 – 2014.

[6] *Systems Engineering Fundamentals*. Department of Defense – Systems Management College, Fort Belvoir, Virginia, Januar 2001. URL `http://www.dau.mil/publications/publicationsDocs/SEFGuide%2001-01.pdf`.

[7] *Risk management guide for DoD acquisition*. Department of Defense, USA, sixth edition, August 2006. URL `http://www.acq.osd.mil/se/docs/2006-RM-Guide-4Aug06-final-version.pdf`.

[8] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering*. Wiley, first edition, 2010.

[9] The Penetration Testing Execution Standard Group. PTES technical guidelines. URL `http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines`. Online; tilgået 15/6-2014. Gruppen består af 20 internationale sikkerhedstestere fra flere brancher.

[10] Luke Herbert. 02233 Network Security – lecture 00 ("introduction"). Slide 27-28, Februar 2014. URL `http://www2.imm.dtu.dk/courses/02233/lectures/Lecture00.pdf`.

[11] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Uncover security design flaws using the STRIDE®-approach. *MSDN Magazine*, November 2006. URL `http://msdn.microsoft.com/en-us/magazine/cc163519.aspx`.

[12] Markus Jakobsson and Susanne Wetzel. *Security Weakness in Bluetooth*, 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.7357`.

[13] Christian Damsgaard Jensen. 02239 Data Security – 02239-1.2 ("An Overview of Computer Security"). Slide 26, September 2013.

[14] Harold Dwight Lasswell. *The structure and function of communication in society*, page 216. Harper and Row, 1948. URL http://www.dhpescu.org/media/elip/Thestructureandfunctionof.pdf.

[15] J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla, and Anandha Murukan. *MSDN Library, Web Development, Ch. 3 (Threat Modeling)*. Microsoft Corporation, Juni 2003. URL http://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011.

[16] Ikuya Morikawa and Yuji Yamaoka. Threat tree templates to ease difficulties in threat modeling. In *2011 International Conference on Network-Based Information Systems, NBiS 2011*, pages 673–678. Fujitsu Laboratories Limited, Kawasaki, Japan, IEEE Computer Society, 2011.

[17] *AccessZone GC530 Serie Adgangskontrolsystem*. MVC-data, November 2013. URL http://www.mvc-data.com/filer/faelles/File/Kort_GC530_Serie.pdf.

[18] *Oracle® Sun Blade X3-2B Security Guide*. Oracle, 2012. URL http://docs.oracle.com/cd/E20881_01/html/E26712/glymd.html. Online; tilgået 30/6 – 2014.

[19] Videnskabens Verden på DR P1. Mange valg forvirrer os. Podcast, Januar 2013. URL http://www.dr.dk/arkivP1/Videnskabensverden/Udsendelser/2013/01/16094807.htm.

[20] Parthajit Panda. The OCTAVE®-approach to information security risk assessment. *ISACA Journal*, 4, 2009. URL http://www.isaca.org/Journal/Past-Issues/2009/Volume-4/Pages/The-OCTAVE-Approach-to-Information-Security-Risk-Assessment1.aspx.

[21] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security In Computing*, chapter 1, pages 1–34. Prentice hall, fourth edition, 2007.

[22] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security In Computing*, chapter 8.2, pages 524–547. Prentice hall, fourth edition, 2007.

[23] OWASP (The Open Web Application Security Project). Threat risk modeling. URL https://www.owasp.org/index.php/Threat_Risk_Modeling. Online; tilgået 2/7 – 2014.

[24] *Application Note S3F80KB – IR remote controller*. Samsung Electronics Co., Ltd., October 2008. URL http://www.samsung.com/global/business/semiconductor/file/product/S3F80KB_RemoteController_AN_REV000_090108-0.pdf. Revision 0.00.

[25] Gary Stoneburner, Clark Hayden, and Alexis Feringa. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security).* NIST, June 2004. URL `http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf`. "NIST Special Publication 800-27 Rev A".